Math. Program., Ser. A (2015) 152:201–245
DOI 10.1007/s10107-014-0784-y

FULL LENGTH PAPER

An adaptive augmented Lagrangian method for large-scale constrained optimization

Frank E. Curtis · Hao Jiang · Daniel P. Robinson

Received: 10 December 2012 / Accepted: 15 April 2014 / Published online: 30 April 2014

Abstract We propose an augmented Lagrangian algorithm for solving large-scale constrained optimization problems. The novel feature of the algorithm is an adaptive update for the penalty parameter motivated by recently proposed techniques for exact penalty methods. This adaptive updating scheme greatly improves the overall performance of the algorithm without sacrificing the strengths of the core augmented Lagrangian framework, such as its ability to be implemented matrix-free. This is important as this feature of augmented Lagrangian methods is responsible for renewed interests in employing such methods for solving large-scale problems. We provide convergence results from remote starting points and illustrate by a set of numerical experiments that our method outperforms traditional augmented Lagrangian methods in terms of critical performance measures.

Keywords Nonlinear optimization · Nonconvex optimization · Large-scale optimization · Augmented Lagrangians · Matrix-free methods · Steering methods

F. E. Curtis—Research supported by National Science Foundation Grant DMS-1016291 and Department of Energy Grant DE-SC0010615
H. Jiang, D.P. Robinson—Research supported by National Science Foundation Grant DMS-1217153.

F. E. Curtis (✉)
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA
e-mail: frank.e.curtis@gmail.com

H. Jiang · D. P. Robinson
Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA
e-mail: hjiang13@jhu.edu

D. P. Robinson
e-mail: daniel.p.robinson@gmail.com

Mathematics Subject Classification (2010) 49M05 · 49M15 · 49M29 · 49M37 · 65K05 · 65K10 · 65K15 · 90C06 · 90C30 · 93B40

1 Introduction

Augmented Lagrangian (AL) methods, also known as methods of multipliers, have been instrumental in the development of algorithms for solving constrained optimization problems since the pioneering works by Hestenes [27] and Powell [37] in the late 1960s. Although subsequently overshadowed by sequential quadratic optimization and interior-point methods in recent decades, AL methods are experiencing a resurgence as interests grow in solving extremely large-scale problems. The attractive features of AL methods in this regard are that they can be implemented matrix-free [1,4,9,32] and possess fast local convergence guarantees under relatively weak assumptions [18,30]. Moreover, certain AL methods—e.g., of the alternating direction variety [21,25]—have proved to be extremely efficient when solving structured large-scale problems [6,38,41].

A critical disadvantage of AL methods when they are applied to solve generic nonlinear problems is that their performance suffers when they are initialized with poor choices of the penalty parameter and/or Lagrange multipliers. Specifically, if the penalty parameter is too large and/or the Lagrange multipliers are poor estimates of the optimal multipliers, then one often finds that little or no progress is made in the primal space due to the iterates veering too far away from the feasible region. This leads to much wasted computational effort, especially in early iterations. (If the constraints are well-scaled, then these issues become less of a practical concern and basic AL methods are quite efficient. Thus, our enhancements are most beneficial for difficult and/or poorly scaled problems.) For example, by using the trivial choice of setting the initial multipliers to zero, one may find that a relatively small value of the penalty parameter is needed before progress is made, which in turn may lead to practical inefficiencies throughout the remainder of the optimization process.

The purpose of this paper is to propose, analyze, and present numerical results for an AL method specifically designed to overcome the disadvantage described in the previous paragraph. We enhance a traditional AL approach with an adaptive penalty parameter update inspired by a recently proposed technique for exact penalty methods [7,8,34]. The adaptive procedure requires that each trial step yields a sufficiently large reduction in linearized constraint violation, thus promoting consistent progress towards constraint satisfaction. We focus on employing our adaptive updating scheme within a trust region method, but a line search algorithm with similar features could be similarly derived.

The paper is organized as follows. In Sect. 2, we outline a traditional AL algorithm to discuss in more detail the inefficiencies that may arise in such an approach. Then, in Sect. 3, we present and analyze our adaptive AL trust region method. We show that the algorithm is well-posed and that it possesses global convergence guarantees. In Sect. 4, we provide numerical results that illustrate the effectiveness of our adaptive penalty parameter updating strategy. Finally, in Sect. 5, we summarize and reflect on our proposed techniques.

Additional background on AL methods can be found in [2,3,5,15,20,22]. We also refer the reader to recent work on stabilized sequential quadratic optimization (SQO) methods [19,31,33,36,40], which share similarly attractive local convergence properties with AL methods. In particular, see [24] for a globally convergent AL method that behaves like a stabilized SQO method near primal solutions. Finally, we comment that other classes of methods use the augmented Lagrangian as a merit function for determining step acceptance and compute trial steps as solutions to subproblems with linear constraints [14,17,23].

Notation We often drop function dependencies once they are defined and use subscripts to denote the iteration number of an algorithm during which a quantity is computed; e.g., we use x_k to denote the kth primal iterate, $f_k := f(x_k)$ to denote the objective value computed at x_k, and similarly for other quantities. We also often use subscripts for constants to indicate the algorithmic quantity to which they correspond; e.g., γ_μ denotes the reduction factor for the parameter μ.

2 A basic augmented Lagrangian algorithm

The algorithm we consider is described in the context of solving the constrained optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) \text{ subject to } c(x) = 0, \quad l \le x \le u, \tag{2.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be twice continuously differentiable. Defining the Lagrangian for (2.1) as

$$\ell(x, y) := f(x) - c(x)^T y,$$

our algorithm seeks a first-order optimal primal-dual solution of (2.1), i.e., an ordered pair (x, y) satisfying

$$0 = F_{\text{OPT}}(x, y) := \begin{pmatrix} F_{\text{L}}(x, y) \\ \nabla_y \ell(x, y) \end{pmatrix} = \begin{pmatrix} P\big[x - \nabla_x \ell(x, y)\big] - x \\ -c(x) \end{pmatrix}, \tag{2.2}$$

where $g(x) := \nabla f(x)$, $J(x) := \nabla c(x)^T$,

$$F_{\text{L}}(x, y) := P[x - \nabla_x \ell(x, y)] - x = P\big[x - \big(g(x) - J(x)^T y\big)\big] - x, \tag{2.3}$$

and P is a projection operator defined componentwise by

$$(P[x])_i = \begin{cases} l_i & \text{if } x_i \le l_i, \\ u_i & \text{if } x_i \ge u_i, \\ x_i & \text{otherwise.} \end{cases}$$

If (2.1) is infeasible, then it is commonly preferred that an algorithm for solving (2.1) returns a stationary point for

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ v(x) \text{ subject to } l \leq x \leq u, \quad \text{where} \quad v(x) := \tfrac{1}{2}\|c(x)\|_2^2, \tag{2.4}$$

namely a point x satisfying

$$0 = F_{\text{FEAS}}(x) := P[x - \nabla_x v(x)] - x = P[x - J(x)^T c(x)] - x. \tag{2.5}$$

If (2.5) holds and $v(x) > 0$, then we say that x is an infeasible stationary point for problem (2.1).

AL methods aim to solve (2.1), or at least (2.4), by solving a sequence of bound-constrained subproblems whose objectives are a weighted sum of the Lagrangian ℓ and the constraint violation measure v. In particular, scaling ℓ by a penalty parameter $\mu \geq 0$, each subproblem involves the minimization of the augmented Lagrangian function

$$\mathcal{L}(x, y, \mu) := \mu\ell(x, y) + v(x) = \mu(f(x) - c(x)^T y) + \tfrac{1}{2}\|c(x)\|_2^2.$$

For future reference, the gradient of the augmented Lagrangian with respect to x evaluated at (x, y, μ) is

$$\nabla_x \mathcal{L}(x, y, \mu) = \mu\big(g(x) - J(x)^T \pi(x, y, \mu)\big), \quad \text{where} \quad \pi(x, y, \mu) := y - \tfrac{1}{\mu}c(x). \tag{2.6}$$

A basic AL algorithm proceeds as follows. Given values for the Lagrange multiplier vector y and penalty parameter μ, the algorithm computes

$$x(y, \mu) := \underset{x\in\mathbb{R}^n}{\text{argmin}}\ \mathcal{L}(x, y, \mu) \text{ subject to } l \leq x \leq u. \tag{2.7}$$

(There may be multiple solutions to the optimization problem in (2.7), or the problem may be unbounded below. However, for simplicity in this discussion, we assume that in (2.7) a point $x(y, \mu)$ can be computed as an approximate stationary point for $\mathcal{L}(\cdot, y, \mu)$.) The first-order optimality condition for the problem in (2.7) is that x yields

$$0 = F_{\text{AL}}\big(x, y, \mu\big) := P[x - \nabla_x \mathcal{L}(x, y, \mu)] - x. \tag{2.8}$$

Inspection of the quantities in (2.2) and (2.8) reveals an important role played by the function π in (2.6). In particular, if $c(x(y, \mu)) = 0$ for $\mu > 0$, then $\pi(x(y, \mu), y, \mu) = y$ and (2.8) implies that $F_{\text{OPT}}(x(y, \mu), y) = 0$, i.e., $(x(y, \mu), y)$ is a first-order optimal solution of (2.1). For this reason, in a basic AL algorithm, if the constraint violation at $x(y, \mu)$ is sufficiently small, then y is set to $\pi(x, y, \mu)$. Otherwise, if the constraint violation is not sufficiently small, then the penalty parameter is decreased to place a higher priority on reducing v during subsequent iterations.

Algorithm 1 outlines a complete AL algorithm. The statement of this algorithm may differ in various ways from previously proposed AL methods, but we claim that the algorithmic structure is a good representation of a generic AL method.

Algorithm 1 Basic Augmented Lagrangian Algorithm

1: Choose constants $\{\gamma_\mu, \gamma_t\} \subset (0, 1)$.
2: Choose an initial primal-dual pair (x_0, y_0) and initialize $\{\mu_0, t_0\} \subset (0, \infty)$.
3: Set $K \leftarrow 0$ and $j \leftarrow 0$.
4: **loop**
5: **if** $F_{\text{OPT}}(x_K, y_K) = 0$, **then**
6: **return** the first-order optimal solution (x_K, y_K).
7: **end if**
8: **if** $\|c_K\|_2 > 0$ and $F_{\text{FEAS}}(x_K) = 0$, **then**
9: **return** the infeasible stationary point x_K.
10: **end if**
11: Compute $x(y_K, \mu_K) \leftarrow \text{argmin}_{x \in \mathbb{R}^n} \mathcal{L}(x, y_K, \mu_K)$ subject to $l \le x \le u$.
12: **if** $\|c(x(y_K, \mu_K))\|_2 \le t_j$, **then**
13: Set $x_{K+1} \leftarrow x(y_K, \mu_K)$.
14: Set $y_{K+1} \leftarrow \pi(x_{K+1}, y_K, \mu_K)$.
15: Set $\mu_{K+1} \leftarrow \mu_K$.
16: Set $t_{j+1} \leftarrow \gamma_t t_j$.
17: Set $j \leftarrow j + 1$.
18: **else**
19: Set $x_{K+1} \leftarrow x_K$ or $x_{K+1} \leftarrow x(y_K, \mu_K)$.
20: Set $y_{K+1} \leftarrow y_K$.
21: Set $\mu_{K+1} \leftarrow \gamma_\mu \mu_K$.
22: **end if**
23: Set $K \leftarrow K + 1$.
24: **end loop**

The technique that we propose in the following section can be motivated by observing a particular drawback of Algorithm 1, namely the manner in which the penalty parameter μ is updated. In Algorithm 1, μ is updated if and only if the **else** clause in line 18 is reached. This is deemed appropriate since after the augmented Lagrangian was minimized in line 11, the constraint violation was larger than the target value t_j; thus, the algorithm decreases μ to place a higher emphasis on reducing v in subsequent iterations. Unfortunately, a side effect of this process is that progress in the primal space based on the update in line 19 is uncertain. Indeed, in such cases, there are typically two possible outcomes. On one hand, the algorithm may set $x_{K+1} \leftarrow x_K$ so that the only result of the iteration—involving the minimization of the (nonlinear) augmented Lagrangian—is that μ is decreased. Alternatively, the algorithm may set $x_{K+1} \leftarrow x(y_K, \mu_K)$ so that progress in the primal-space may be obtained, but not necessarily; indeed, in some cases this update may be counterproductive. In this case, the only certain progress made during the iteration is the decrease of μ.

The scenario described in the previous paragraph illustrates that a basic AL algorithm may be very inefficient, especially during early iterations when the penalty parameter μ may be too large or the multiplier y is a poor estimate of the optimal multiplier vector. The method that we propose in the following section is designed to overcome this potential inefficiency by adaptively updating the penalty parameter *during* the minimization process for the augmented Lagrangian.

We close this section by noting that the minimization in line 11 of Algorithm 1 is itself an iterative process, the iterations of which we refer to as "minor" iterations. This is our motivation for using K as the "major" iteration counter, so as to distinguish it from the iteration counter k used in our method, which is similar—e.g., in terms of computational cost—to the "minor" iterations in Algorithm 1.

3 An adaptive augmented Lagrangian trust region algorithm

In this section, we propose and analyze an AL trust region method with an adaptive updating scheme for the penalty parameter. The new key idea is to measure the improvement towards linearized constraint satisfaction obtained by a given trial step and compare it to that obtained by a step that solely seeks feasibility. If the former improvement is not sufficiently large compared to the latter and the current constraint violation is not sufficiently small, then the penalty parameter is decreased to place a higher emphasis on minimizing constraint violation during the current iteration.

Our strategy involves a set of easily implementable conditions designed around the following models of the constraint violation measure, Lagrangian, and augmented Lagrangian, respectively:

$$q_v(s; x) := \tfrac{1}{2}\|c(x) + J(x)s\|_2^2 \approx v(x + s); \tag{3.1}$$

$$q_\ell(s; x, y) := \ell(x, y) + \nabla_x \ell(x, y)^T s + \tfrac{1}{2} s^T \nabla_{xx}^2 \ell(x, y)s \approx \ell(x + s, y); \tag{3.2}$$

$$q(s; x, y, \mu) := \mu q_\ell(s; x, y) + q_v(s; x) \approx \mathcal{L}(x + s, y, \mu). \tag{3.3}$$

We remark that this approximation to the augmented Lagrangian is not the standard second-order Taylor series approximation; instead, we employ a Gauss-Newton model for the constraint violation measure.

Each iteration of our algorithm requires the computation of a trial step s_k toward minimizing the augmented Lagrangian. Ideally, this trial step will also make progress toward solving (2.1) and, in particular, toward minimizing v. To promote this behavior, we compute a step s_k that predicts a decrease in the augmented Lagrangian as well as an acceptable value of linearized constraint violation.

Whether a computed step s_k yields an acceptable value of linearized constraint violation from the current iterate x_k depends on that yielded by a *steering* step r_k, defined as an approximate solution of

$$\underset{r \in \mathbb{R}^n}{\text{minimize}}\ q_v(r; x_k) \text{ subject to } l \leq x_k + r \leq u,\ \|r\|_2 \leq \theta_k, \tag{3.4}$$

where $\delta > 0$ is a constant and both

$$\theta_k := \theta(x_k, \delta_k) := \min\{\delta_k, \delta \|F_{\text{FEAS}}(x_k)\|_2\} \geq 0 \tag{3.5}$$

and $\delta_k > 0$ are set dynamically within our algorithm. (Note that a consequence of this choice of trust-region radius θ_k in (3.4) is that it approaches zero as the algorithm approaches stationary points of the constraint violation measure [39]. This keeps the

steering step from being too large relative to the progress that can be made toward minimizing v. Moreover, since in practice δ is chosen to be large, the term $\delta\|F_{\text{FEAS}}(x_k)\|_2$ will not impede superlinear convergence when the inverse of the Hessian matrix is smaller in norm than δ in the neighborhood of solutions.) Importantly, we allow for inexact solutions to this subproblem that still ensure convergence since we are interested in matrix-free implementations of our methods; see (3.11b). To this end, we compute a Cauchy step for subproblem (3.4) as

$$\bar{r}_k := \bar{r}(x_k, \theta_k) := P[x_k - \bar{\beta}_k J_k^T c_k] - x_k \tag{3.6}$$

such that $\bar{r}_k$ satisfies

$$\Delta q_v(\bar{r}_k; x_k) := q_v(0; x_k) - q_v(\bar{r}_k; x_k) \geq -\varepsilon_r \bar{r}_k^T J_k^T c_k \ \text{ and } \ \|\bar{r}_k\|_2 \leq \theta_k \tag{3.7}$$

for some $\bar{\beta}_k := \bar{\beta}(x_k, \theta_k)$ and $\varepsilon_r \in (0, 1)$. Appropriate values for $\bar{\beta}_k$, $\bar{r}_k$, and the auxiliary nonnegative scalar quantities ε_k and Γ_k (to be used shortly) may be computed from Algorithm 2.

Algorithm 2 Cauchy step computation for the feasibility subproblem (3.4)

procedure CAUCHY_FEASIBILITY(x_k, θ_k)
  restrictions : $\theta_k \geq 0$.
  available constants : $\{\varepsilon_r, \gamma\} \subset (0, 1)$.
  Compute l_k as the smallest nonnegative integer satisfying $\|P[x_k - \gamma^{l_k} J_k^T c_k] - x_k\|_2 \leq \theta_k$
  if $l_k > 0$ **then**
    Set $\Gamma_k \leftarrow \min\{2, \frac{1}{2}(1 + \|P[x_k - \gamma^{l_k - 1} J_k^T c_k] - x_k\|_2/\theta_k)\}$.
  else
    Set $\Gamma_k \leftarrow 2$.
  end if
  Set $\bar{\beta}_k \leftarrow \gamma^{l_k}$, $\bar{r}_k \leftarrow P[x_k - \bar{\beta}_k J_k^T c_k] - x_k$, and $\varepsilon_k \leftarrow 0$.
  while $\bar{r}_k$ does not satisfy (3.7) **do**
    Set $\varepsilon_k \leftarrow \max(\varepsilon_k, -\Delta q_v(\bar{r}_k; x_k)/\bar{r}_k^T J_k^T c_k)$.
    Set $\bar{\beta}_k \leftarrow \gamma \bar{\beta}_k$ and $\bar{r}_k \leftarrow P[x_k - \bar{\beta}_k J_k^T c_k] - x_k$.
  end while
  return : $(\bar{\beta}_k, \bar{r}_k, \varepsilon_k, \Gamma_k)$
end procedure

The predicted reduction in the constraint violation from x_k yielded by $\bar{r}_k$ as measured by $\Delta q_v(\bar{r}_k; x_k)$ is guaranteed to be positive at any x_k that is not first-order critical for v under the bound constraints; see part (i) of Lemma 3.5. The reduction $\Delta q_v(r_k; x_k)$ is defined similarly for the steering step r_k, whose computation will be discussed later in this section.

With the Cauchy step for our steering problem computed, we proceed to identify a new penalty parameter μ_k and trial step s_k that satisfy certain properties. Specifically, the step s_k is defined as an approximate solution of the following quadratic trust-region subproblem:

$$\underset{s \in \mathbb{R}^n}{\text{minimize}}\ q(s; x_k, y_k, \mu_k) \text{ subject to } l \leq x_k + s \leq u, \ \|s\|_2 \leq \Theta_k, \tag{3.8}$$

where

$$\Theta_k := \Theta(x_k, y_k, \mu_k, \delta_k, \Gamma_k) = \Gamma_k \min\{\delta_k, \delta \|F_{\text{AL}}(x_k, y_k, \mu_k)\|_2\} \geq 0 \tag{3.9}$$

with $\Gamma_k > 1$ returned from Algorithm 2. Similar to (3.4), this definition of the trust-region radius involves the first-order optimality measure F_{AL} for minimizing $\mathcal{L}(\cdot, y_k, \mu_k)$. This choice ensures that the trust-region radius Θ_k is driven to zero as first-order minimizers of $\mathcal{L}(\cdot, y_k, \mu_k)$ are approached.

Problem (3.8) is, in fact, a quadratic optimization problem if we used the ℓ_∞-norm to define the trust-region constraint. Nonetheless, our algorithm uses the ℓ_2-norm and allows for inexact solutions to this subproblem that still ensure convergence; see (3.11a). We define the Cauchy step for problem (3.8) as

$$\bar{s}_k := \bar{s}(x_k, y_k, \mu_k, \Theta_k, \varepsilon_k) := P[x_k - \bar{\alpha}_k \nabla_x \mathcal{L}(x_k, y_k, \mu_k)] - x_k,$$

such that $\bar{s}_k$ yields

$$\begin{aligned} \Delta q(\bar{s}_k; x_k, y_k, \mu_k) &:= q(0; x_k, y_k, \mu_k) - q(\bar{s}_k; x_k, y_k, \mu_k) \\ &\geq -\frac{(\varepsilon_k + \varepsilon_r)}{2} \bar{s}_k^T \nabla_x \mathcal{L}(x_k, y_k, \mu_k) \text{ and } \|\bar{s}_k\|_2 \leq \Theta_k \end{aligned} \tag{3.10}$$

for some $\bar{\alpha}_k = \bar{\alpha}(x_k, y_k, \mu_k, \Theta_k, \varepsilon_k)$, where $\varepsilon_k \geq 0$ is returned from Algorithm 2. Appropriate values for $\bar{\alpha}_k$ and $\bar{s}_k$ may be computed from Algorithm 3. (The importance of using Γ_k in (3.9) and ε_k in (3.10) may be seen in the proofs of Lemmas 3.3 and 3.4 in Sect. 3.1.)

Algorithm 3 Cauchy step computation for the AL subproblem (3.8).

procedure CAUCHY_AL($x_k, y_k, \mu_k, \Theta_k, \varepsilon_k$)
  restrictions : $\mu_k > 0$, $\Theta_k > 0$, and $\varepsilon_k \geq 0$.
  available constant : $\gamma \in (0, 1)$.
  Set $\bar{\alpha}_k \leftarrow 1$ and $\bar{s}_k \leftarrow P[x_k - \bar{\alpha}_k \nabla_x \mathcal{L}(x_k, y_k, \mu_k)] - x_k$.
  while (3.10) is not satisfied **do**
    Set $\bar{\alpha}_k \leftarrow \gamma \bar{\alpha}_k$ and $\bar{s}_k \leftarrow P[x_k - \bar{\alpha}_k \nabla_x \mathcal{L}(x_k, y_k, \mu_k)] - x_k$.
  end while
  return : $(\bar{\alpha}_k, \bar{s}_k)$
end procedure

The predicted reduction in $\mathcal{L}(\cdot, y_k, \mu_k)$ from x_k yielded by the step $\bar{s}_k$ and measured by $\Delta q(\bar{s}_k; x_k, y_k, \mu_k)$ is guaranteed to be positive at any x_k that is not first-order critical for $\mathcal{L}(\cdot, y_k, \mu_k)$ under the bound constraints; see part (ii) of Lemma 3.5 for a more precise lower bound for this predicted change. The reduction $\Delta q(s_k; x_k, y_k, \mu_k)$ is defined similarly for the trial step s_k.

We now describe the kth iteration of our algorithm, specified as Algorithm 4 on page 8. Let (x_k, y_k) be the current primal-dual iterate. We begin by checking whether (x_k, y_k) is a first-order optimal point for (2.1) or if x_k is an infeasible stationary point, and terminate in either case. Otherwise, we enter the **while** loop in line 11 to obtain a value for the penalty parameter for which $F_{\text{AL}}(x_k, y_k, \mu_k) \neq 0$; recall (2.8).

This is appropriate as the purpose of each iteration is to compute a step towards a bound-constrained minimizer of the augmented Lagrangian $\mathcal{L}(\cdot, y_k, \mu_k)$, and if $F_{\text{AL}}(x_k, y_k, \mu_k) = 0$, then no feasible descent directions for $\mathcal{L}(\cdot, y_k, \mu_k)$ from x_k exist. (Lemma 3.2 shows that this **while** loop terminates finitely.) Next, we enter a **while** loop on line 19 that recovers an approximate solution r_k to problem (3.4) and an approximate solution s_k to problem (3.8) that satisfy

$$\begin{aligned}
&\Delta q(s_k; x_k, y_k, \mu_k) \\
&\quad \geq \kappa_1 \Delta q(\bar{s}_k; x_k, y_k, \mu_k) > 0, \quad l \leq x_k + s_k \leq u, \quad \|s_k\|_2 \leq \Theta_k, && \text{(3.11a)} \\
&\quad \Delta q_v(r_k; x_k) \geq \kappa_2 \Delta q_v(\bar{r}_k; x_k), \ l \leq x_k + r_k \leq u, \quad \|r_k\|_2 \leq \theta_k, && \text{(3.11b)} \\
&\text{and} \ \ \Delta q_v(s_k; x_k) \geq \min\{\kappa_3 \Delta q_v(r_k; x_k), v_k - \tfrac{1}{2}(\kappa_t t_j)^2\}, && \text{(3.11c)}
\end{aligned}$$

where $\{\kappa_1, \kappa_2, \kappa_3, \kappa_t\} \subset (0, 1)$ and the quantity $t_j > 0$ represents the jth target for the constraint violation. At this stage of the algorithm, there are many vectors r_k and s_k that satisfy (3.11), but for our purposes we simply prove that they are satisfied for $r_k = \bar{r}_k$ and $s_k = \bar{s}_k$; see Theorem 3.6.

The conditions in (3.11) can be motivated as follows. Conditions (3.11a) and (3.11b) ensure that the trial step s_k and steering step r_k yield nontrivial decreases in the models of the augmented Lagrangian and the constraint violation, respectively, compared to their Cauchy points. The motivation for condition (3.11c) is more complex as it involves a minimum of two values on the right-hand side, but this condition is critical as it ensures that the reduction in the constraint violation model is sufficiently large for the trial step. The first quantity on the right-hand side, if it were the minimum of the two, would require the decrease in the model q_v yielded by s_k to be a fraction of that obtained by the steering step r_k; see [7,8] for similar conditions enforced in exact penalty methods. The second quantity is the difference between the current constraint violation and a measure involving a fraction of the target value $t_j > 0$. Note that this second term allows the minimum to be negative. Therefore, this condition allows for the trial step s_k to predict an increase in the constraint violation, but only if the current constraint violation is sufficiently within the target value t_j. It is worthwhile to note that in general one may consider allowing the penalty parameter to increase as long as the resulting trial step satisfies conditions (3.11a)–(3.11c) and the parameter eventually settles down at a small enough value to ensure that constraint violation is minimized. However, as this only would be a heuristic and not theoretically interesting, we ignore this possibility and simply have the parameter decrease monotonically. We remark that the computation of r_k requires extra effort beyond that for computing s_k, but the expense is minor as r_k can be computed in parallel with s_k and must only satisfy a Cauchy decrease condition [i.e., (3.11b)] for (3.4).

With the trial step s_k in hand, we proceed to compute the ratio

$$\rho_k \leftarrow \frac{\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_k + s_k, y_k, \mu_k)}{\Delta q(s_k; x_k, y_k, \mu_k)} \tag{3.12}$$

Algorithm 4 Adaptive Augmented Lagrangian Trust Region Algorithm

Choose $\{\gamma, \gamma_\mu, \gamma_t, \gamma_T, \gamma_\delta, \kappa_F, \kappa_1, \kappa_2, \kappa_3, \varepsilon_r, \kappa_t, \eta_s, \eta_{vs}\} \subset (0, 1)$, $\{\delta, \delta_R, \epsilon, Y\} \subset (0, \infty)$, $\Gamma_\delta > 1$ such that $\eta_{vs} \geq \eta_s$.
Choose initial primal-dual pair (x_0, y_0) and initialize $\{\mu_0, \delta_0, t_0, t_1, T_1, Y_1\} \subset (0, \infty)$ such that $Y_1 \geq Y$ and $\|y_0\|_2 \leq Y_1$.
Set $k \leftarrow 0$, $k_0 \leftarrow 0$, and $j \leftarrow 1$.
loop
  if $F_{\text{OPT}}(x_k, y_k) = 0$, **then**
    return the first-order optimal solution (x_k, y_k).
  end if
  if $\|c_k\|_2 > 0$ and $F_{\text{FEAS}}(x_k) = 0$, **then**
    return the infeasible stationary point x_k.
  end if
  while $F_{\text{AL}}(x_k, y_k, \mu_k) = 0$, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$.
  end while
  Define θ_k by (3.5).
  Use Algorithm 2 to compute $(\overline{\beta}_k, \overline{r}_k, \varepsilon_k, \Gamma_k) = \text{CAUCHY\_FEASIBILITY}(x_k, \theta_k)$.
  Define Θ_k by (3.9).
  Use Algorithm 3 to compute $(\overline{\alpha}_k, \overline{s}_k) = \text{CAUCHY\_AL}(x_k, y_k, \mu_k, \Theta_k, \varepsilon_k)$.
  Compute approximate solutions r_k to (3.4) and s_k to (3.8) that satisfy (3.11a)–(3.11b).
  while (3.11c) is not satisfied or $F_{\text{AL}}(x_k, y_k, \mu_k) = 0$, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$ and define Θ_k by (3.9).
    Use Algorithm 3 to compute $(\overline{\alpha}_k, \overline{s}_k) = \text{CAUCHY\_AL}(x_k, y_k, \mu_k, \Theta_k, \varepsilon_k)$.
    Compute an approximate solution s_k to (3.8) satisfying (3.11a).
  end while
  Compute ρ_k from (3.12).
  if $\rho_k \geq \eta_{vs}$, **then**
    Set $x_{k+1} \leftarrow x_k + s_k$ and $\delta_{k+1} \leftarrow \max\{\delta_R, \Gamma_\delta \delta_k\}$. ▷ very successful iteration
  else if $\rho_k \geq \eta_s$, **then**
    Set $x_{k+1} \leftarrow x_k + s_k$ and $\delta_{k+1} \leftarrow \max\{\delta_R, \delta_k\}$. ▷ successful iteration
  else
    Set $x_{k+1} \leftarrow x_k$ and $\delta_{k+1} \leftarrow \gamma_\delta \delta_k$. ▷ unsuccessful iteration
  end if
  if $\|c_{k+1}\|_2 \leq t_j$, **then**
    Compute any $\widehat{y}_{k+1}$ satisfying (3.13).
    if $\min\{\|F_{\text{L}}(x_{k+1}, \widehat{y}_{k+1})\|_2, \|F_{\text{AL}}(x_{k+1}, y_k, \mu_k)\|_2\} \leq T_j$, **then**
      Set $k_j \leftarrow k + 1$ and $Y_{j+1} \leftarrow \max\{Y, t_{j-1}^{-\epsilon}\}$.
      Set $t_{j+1} \leftarrow \min\{\gamma_t t_j, t_j^{1+\epsilon}\}$ and $T_{j+1} \leftarrow \gamma_T \min\{1, \mu_k\} T_j$.
      Set y_{k+1} from (3.14) where α_y satisfies (3.15).
      Set $j \leftarrow j + 1$.
    else
      Set $y_{k+1} \leftarrow y_k$.
    end if
  else
    Set $y_{k+1} \leftarrow y_k$.
  end if
  Set $\mu_{k+1} \leftarrow \mu_k$.
  Set $k \leftarrow k + 1$.
end loop

of actual-to-predicted decrease in $\mathcal{L}(\cdot, y_k, \mu_k)$. Since $\Delta q(s_k; x_k, y_k, \mu_k)$ is positive by (3.11a), it follows that if $\rho_k \geq \eta_s$ for $\eta_s \in (0, 1)$, then the augmented Lagrangian has been sufficiently reduced. In such cases, we accept $x_k + s_k$ as the next iterate.

Moreover, if we find that $\rho_k \geq \eta_{vs}$ for $\eta_{vs} \in (\eta_s, 1)$, then our choice of trust region radius may have been overly cautious so we multiply the upper bound for the trust region radius (i.e., δ_k) by $\Gamma_\delta > 1$. If $\rho_k < \eta_s$, then the trust region radius may have been too large, so we counter this by multiplying the upper bound by $\gamma_\delta \in (0, 1)$.

Next we determine how to define our next multiplier vector y_{k+1}. The first condition that we check is whether the constraint violation at x_{k+1} is sufficiently small compared to t_j. If this requirement is met, then we may compute any prospective multiplier estimate $\widehat{y}_{k+1}$ that satisfies

$$\|F_{\text{L}}(x_{k+1}, \widehat{y}_{k+1})\|_2 \leq \min\left\{\|F_{\text{L}}(x_{k+1}, y_k)\|_2, \|F_{\text{L}}(x_{k+1}, \pi(x_{k+1}, y_k, \mu_k))\|_2\right\}. \quad (3.13)$$

Computing $\widehat{y}_{k+1}$ as an approximate least-squares multiplier estimate from an associated linear optimization problem or simply from π as defined in (2.6) are both viable options, but for flexibility in the statement of our algorithm we simply enforce (3.13). With it being satisfied, we then check if either $\|F_{\text{L}}(x_{k+1}, \widehat{y}_{k+1})\|_2$ or $\|F_{\text{AL}}(x_{k+1}, y_k, \mu_k)\|_2$ is sufficiently small with respect to a target value $T_j > 0$. If this condition is satisfied, we choose new target values $t_{j+1} < t_j$ and $T_{j+1} < T_j$, and then set $Y_{j+1} \geq Y_j$ and

$$y_{k+1} \leftarrow (1 - \alpha_y) y_k + \alpha_y \widehat{y}_{k+1}, \quad (3.14)$$

where α_y is the largest value in $[0, 1]$ such that

$$\|(1 - \alpha_y) y_k + \alpha_y \widehat{y}_{k+1}\|_2 \leq Y_{j+1}. \quad (3.15)$$

Note that this updating procedure is well-defined since the choice $\alpha_y \leftarrow 0$ results in $y_{k+1} \leftarrow y_k$, which at least means that (3.15) is satisfiable by this α_y. On the other-hand, i.e., when the aforementioned condition is not satisfied, we simply set $y_{k+1} \leftarrow y_k$.

For future reference, we define the subset of iterations where line 35 of Algorithm 4 is reached as

$$\mathcal{Y} := \left\{k_j : \|c_{k_j}\|_2 \leq t_j \text{ and } \min\{\|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2, \|F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})\|_2\} \leq T_j\right\}. \quad (3.16)$$

We conclude this section by noting that, in practice, lines 5 and 8 in Algorithm 4 should be replaced by practical conditions that include positive stopping tolerances. For example, see the implementation details given in Sect. 4.1.

3.1 Well-posedness

We prove that Algorithm 4 is well-posed—i.e., either the algorithm will terminate finitely or will produce an infinite sequence $\{(x_k, y_k, \mu_k)\}_{k \geq 0}$ of iterates—under the following assumption. This assumption is assumed throughout this section and is therefore not stated explicitly in each result.

Assumption 3.1 *At a given x_k, the objective function f and constraint function c are both twice-continuously differentiable.*

Well-posedness in our context first requires that the **while** loop that begins at line 11 of Algorithm 4 terminates finitely. The proof of this fact requires the following simple result.

Lemma 3.1 *Suppose $l \leq x \leq u$ and let v be any vector in $\mathbb{R}^n$. If there exists a scalar $\xi_s > 0$ such that $P[x - \xi_s v] = x$, then $P[x - \xi v] = x$ for all $\xi > 0$.*

Proof Since $l \leq x \leq u$, it follows the definition of the projection operator P and the facts that $\xi_s > 0$ and $P[x - \xi_s v] = x$ that

$$v_i \geq 0 \ \text{ if } \ x_i = l_i; \ \ v_i \leq 0 \ \text{ if } \ x_i = u_i; \ \ \text{and } \ v_i = 0 \ \text{ otherwise.}$$

It follows that $P[x - \xi v] = x$ for all $\xi > 0$, as desired. □

Lemma 3.2 *If line 11 of Algorithm 4 is reached, then $F_{\text{AL}}(x_k, y_k, \mu) \neq 0$ for all sufficiently small $\mu > 0$.*

Proof Suppose that line 11 is reached and, to reach a contradiction, suppose also that there exists an infinite positive sequence $\{\xi_h\}_{h \geq 0}$ such that $\xi_h \to 0$ and

$$F_{\text{AL}}(x_k, y_k, \xi_h) = P\left[x_k - \xi_h(g_k - J_k^T y_k) - J_k^T c_k\right] - x_k = 0 \ \text{ for all } \ h \geq 0. \quad (3.17)$$

It follows from (3.17) and the fact that $\xi_h \to 0$ that

$$F_{\text{FEAS}}(x_k) = P[x_k - J_k^T c_k] - x_k = 0.$$

If $c_k \neq 0$, then Algorithm 4 would have terminated in line 9; hence, since line 11 is reached, we must have $c_k = 0$. We then may conclude from (3.17), the fact that $\{\xi_h\}$ is positive for all h, and Lemma 3.1 that $F_{\text{L}}(x_k, y_k) = 0$. Combining this with (2.2) and the fact that $c_k = 0$, it follows that $F_{\text{OPT}}(x_k, y_k) = 0$ so that (x_k, y_k) is a first-order optimal point for (2.1). However, under these conditions, Algorithm 4 would have terminated in line 6. Hence, we have a contradiction to the existence of the sequence $\{\xi_h\}$. □

We now show that the Cauchy step computations given by Algorithms 2 and 3 are well defined when called in steps 15, 17, and 21 of Algorithm 4.

Lemma 3.3 *The following hold true:*

(i) The computation of $(\overline{\beta}_k, \overline{r}_k, \varepsilon_k, \Gamma_k)$ in step 15 is well defined and yields $\Gamma_k \in (1, 2]$ and $\varepsilon_k \in [0, \varepsilon_r)$.
(ii) The computation of $(\overline{\alpha}_k, \overline{s}_k)$ in steps 17 and 21 is well defined.

Proof We start by proving part (i). Consider the call to Algorithm 2 made during step 15 of Algorithm 4. If $\theta_k = 0$, then, since $\delta_k > 0$ by construction, we must have $F_{\text{FEAS}}(x_k) = 0$, which in turn implies that Algorithm 2 trivially computes $l_k = 0$,

$\Gamma_k = 2$, $\overline{\beta}_k = 1$, and $\varepsilon_k = 0$. In this case, (i) clearly holds, so now let us suppose that $\theta_k > 0$. If $l_k = 0$, then $\Gamma_k = 2$; otherwise, if $l_k > 0$, then it follows that either $\Gamma_k = 2$ or

$$2 > \Gamma_k = \tfrac{1}{2}\left(1 + \frac{\|P\left[x_k - \gamma^{l_k - 1} J_k^T c_k\right] - x_k\|_2}{\theta_k}\right) > \tfrac{1}{2}\left(1 + \frac{\theta_k}{\theta_k}\right) = 1.$$

Next, the **while** loop at line 11 terminates finitely as shown by [35, Theorem 4.2] since $\varepsilon_r \in (0, 1)$. The fact that $\varepsilon_k \in [0, \varepsilon_r)$ holds since $\varepsilon_r \in (0, 1)$ by choice, ε_k is initialized to zero in Algorithm 2, and every time ε_k is updated we have $-\Delta q_v(\overline{r}_k; x_k)/\overline{r}_k^T J_k^T c_k < \varepsilon_r$ (by the condition of the **while** loop).

Now consider part (ii). Regardless of how step 17 or 21 is reached in Algorithm 4, we have that $\mu_k > 0$ and $\Theta_k \geq 0$ by construction, and from part (i) we have that $\Gamma_k \in (1, 2]$ and $\varepsilon_k \in [0, \varepsilon_r)$. If $\Theta_k = 0$, then, since $\delta_k > 0$ by construction, it follows that $F_{\text{AL}}(x_k, y_k, \mu_k) = 0$, and therefore Algorithm 3 terminates with $\overline{\alpha}_k = 1$ and $\overline{s}_k = 0$. Thus, we may continue supposing that $\Theta_k > 0$. We may now use the fact that

$$0 < \varepsilon_r/2 \leq (\varepsilon_k + \varepsilon_r)/2 < \varepsilon_r < 1$$

and [35, Theorem 4.2] to say that Algorithm 3 will terminate finitely with $(\overline{\alpha}_k, \overline{s}_k)$ satisfying (3.10). □

The following result illustrates critical relationships between the quadratic models q_v and q as $\mu \to 0$.

Lemma 3.4 *Let* $(\overline{\beta}_k, \overline{r}_k, \varepsilon_k, \Gamma_k) \leftarrow$ CAUCHY_FEASIBILITY(x_k, θ_k) *with* θ_k *defined by* (3.5) *and let* $(\overline{\alpha}_k(\mu), \overline{s}_k(\mu)) \leftarrow$ CAUCHY_AL$(x_k, y_k, \mu, \Theta_k(\mu), \varepsilon_k)$ *with* $\Theta_k(\mu) := \Gamma_k \min\{\delta_k, \delta \|F_{\text{AL}}(x_k, y_k, \mu)\|_2\}$ *(see* (3.9)*). Then, the following hold true:*

$$\lim_{\mu \to 0} \left(\max_{\|s\|_2 \leq 2\delta_k} |q(s; x_k, y_k, \mu) - q_v(s; x_k)| \right) = 0, \tag{3.18a}$$

$$\lim_{\mu \to 0} \nabla_x \mathcal{L}(x_k, y_k, \mu) = J_k^T c_k, \tag{3.18b}$$

$$\lim_{\mu \to 0} \overline{s}_k(\mu) = \overline{r}_k, \tag{3.18c}$$

$$\textit{and} \quad \lim_{\mu \to 0} \Delta q_v(\overline{s}_k(\mu); x_k) = \Delta q_v(\overline{r}_k; x_k). \tag{3.18d}$$

Proof Since x_k and y_k are fixed, for the purposes of this proof we drop them from all function dependencies. From the definitions of q and q_v, it follows that for some $M > 0$ independent of μ we have

$$\max_{\|s\|_2 \leq 2\delta_k} |q(s; \mu) - q_v(s)| = \mu \max_{\|s\|_2 \leq 2\delta_k} |q_\ell(s)| \leq \mu M.$$

Hence, (3.18a) follows. Similarly, we have

$$\nabla_x \mathcal{L}(\mu) - J_k^T c_k = \mu(g_k - J_k^T y_k),$$

from which it is clear that (3.18b) holds.

We now show (3.18c) by considering two cases. We emphasize that throughout these two arguments all quantities in Algorithm 2 are unaffected by μ, so the reader can consider them as fixed.

Case 1: Suppose that $F_{\text{FEAS}}(x_k) = 0$. This implies that $\theta_k = \min\{\delta_k, \delta \| F_{\text{FEAS}}(x_k)\|_2\} = 0$, so that $\bar{r}_k = 0$ and $\Delta q_v(\bar{r}_k) = 0$. Moreover, from (3.18b) we have $\Theta_k(\mu) \to 0$ as $\mu \to 0$, which means that $\bar{s}_k(\mu) \to 0 = \bar{r}_k$.

Case 2: Suppose $F_{\text{FEAS}}(x_k) \neq 0$. We find it useful to define the functions

$$r_k(l) = P[x_k - \gamma^l J_k^T c_k] - x_k \quad \text{and} \quad s_k(l, \mu) = P[x_k - \gamma^l \nabla_x \mathcal{L}(\mu)] - x_k$$

for any integer $l \geq 0$ and scalar $\mu > 0$. We also let $l_\beta \geq 0$ be the integer such that $\bar{\beta}_k = \gamma^{l_\beta}$, which implies

$$\bar{r}_k = r_k(l_\beta). \tag{3.19}$$

It follows from (3.18b) that

$$\lim_{\mu \to 0} s_k(l, \mu) = r_k(l) \quad \text{for any} \quad l \geq 0 \quad \text{and, in particular,} \quad \lim_{\mu \to 0} s_k(l_\beta, \mu) = r_k(l_\beta) = \bar{r}_k. \tag{3.20}$$

Therefore, to show (3.18c), it suffices to prove that

$$\bar{s}_k(\mu) = s_k(l_\beta, \mu) \quad \text{for all} \quad \mu > 0 \quad \text{sufficiently small.} \tag{3.21}$$

Since the Cauchy computation for $\bar{s}_k(\mu)$ computes a nonnegative integer $l_{\alpha,\mu}$ so that

$$\bar{s}_k(\mu) = P\left[x_k - \gamma^{l_{\alpha,\mu}} \nabla_x \mathcal{L}(\mu)\right] - x_k,$$

to prove (3.21) it suffices to show that $l_{\alpha,\mu} = l_\beta$ for all $\mu > 0$ sufficiently small. Before proving this result, however, we show that

$$\min\left(l_\beta, l_{\alpha,\mu}\right) \geq l_k \quad \text{for all} \quad \mu > 0 \quad \text{sufficiently small,} \tag{3.22}$$

where l_k is computed in Algorithm 2. If $l_k = 0$, then (3.22) holds trivially. Thus, let us suppose that $l_k > 0$. We may first observe that the inequality $l_\beta \geq l_k$ holds by construction of Algorithm 2. Also, it follows from the definition of $\Theta_k(\mu)$, (3.18b), the definition of Γ_k, $\theta_k > 0$, and the fact that $\| P[x_k - \gamma^{l_k - 1} J_k^T c_k] - x_k \|_2 > \theta_k$ due to the choice of l_k in Algorithm 2, that

$$\begin{aligned}\lim_{\mu\to 0}\Theta_k(\mu) &= \lim_{\mu\to 0}\Gamma_k\min(\delta_k,\delta\|F_{\text{AL}}(x_k,y_k,\mu)\|_2) = \Gamma_k\min(\delta_k,\delta\|F_{\text{FEAS}}(x_k)\|_2)\\ &= \Gamma_k\theta_k\\ &= \min\left[2,\tfrac{1}{2}\left(1+\frac{\|P[x_k-\gamma^{l_k-1}J_k^Tc_k]-x_k\|_2}{\theta_k}\right)\right]\theta_k\\ &= \min\left[2\theta_k,\tfrac{1}{2}\left(\theta_k+\|P[x_k-\gamma^{l_k-1}J_k^Tc_k]-x_k\|_2\right)\right]\\ &\in \left(\theta_k,\ \|P[x_k-\gamma^{l_k-1}J_k^Tc_k]-x_k\|_2\right).\end{aligned}$$

Using this and (3.18b), we may observe that

$$\begin{aligned}\lim_{\mu\to 0}\|P[x_k-\gamma^{l_k-1}\nabla_x\mathcal{L}(\mu)]-x_k\|_2 &= \|P[x_k-\gamma^{l_k-1}J_k^Tc_k]-x_k\|_2\\ &> \Theta_k(\mu)\quad\text{for all}\quad\mu>0\quad\text{sufficiently small,}\end{aligned}$$

which shows that $l_{\alpha,\mu}\geq l_k$ for all $\mu>0$ sufficiently small and, consequently, that (3.22) again holds.

We now proceed to prove that $l_{\alpha,\mu}=l_\beta$ for all $\mu>0$ sufficiently small. It follows from the definition of l_β above, (3.19), the structure of Algorithm 2, definition of ε_k, and part (i) of Lemma 3.3 that

$$\begin{aligned}-\frac{\Delta q_v(\bar r_k)}{\bar r_k^TJ_k^Tc_k} &= -\frac{\Delta q_v\big(r_k(l_\beta)\big)}{r_k(l_\beta)^TJ_k^Tc_k}\geq\epsilon_r\quad\text{and}\\ -\frac{\Delta q_v\big(r_k(l)\big)}{r_k(l)^TJ_k^Tc_k} &\leq\epsilon_k<\epsilon_r\quad\text{for all integers}\quad l_k\leq l<l_\beta.\end{aligned}\tag{3.23}$$

(Note that [11, Theorem 12.1.4] shows that all denominators in (3.23) are negative.) It follows from (3.18b), (3.20), (3.18a), (3.23), and part (i) of Lemma 3.3 that

$$\lim_{\mu\to 0}-\frac{\Delta q\big(s_k(l_\beta,\mu)\big)}{s_k(l_\beta,\mu)^T\nabla_x\mathcal{L}(\mu)}=-\frac{\Delta q_v(\bar r_k)}{\bar r_k^TJ_k^Tc_k}\geq\varepsilon_r>\frac{\varepsilon_k+\varepsilon_r}{2}\tag{3.24}$$

and

$$\begin{aligned}\lim_{\mu\to 0}-\frac{\Delta q\big(s_k(l,\mu)\big)}{s_k(l,\mu)^T\nabla_x\mathcal{L}(\mu)} &= -\frac{\Delta q_v\big(r_k(l)\big)}{r_k(l)^TJ_k^Tc_k}\\ &\leq\varepsilon_k<\frac{\varepsilon_k+\varepsilon_r}{2}\quad\text{for all integers}\quad l_k\leq l<l_\beta.\end{aligned}\tag{3.25}$$

It now follows from (3.22), (3.24), (3.25), and (3.10) that $l_{\alpha,\mu}=l_\beta$ for all $\mu>0$ sufficiently small, which proves (3.18c).

Finally, we notice that (3.18d) follows from (3.18c) and continuity of the model q_v. □

To show that Algorithm 4 is well-posed, we also need the following results.

Lemma 3.5 *Let Ω be any value such that*

$$\Omega \geq \max\{\|\mu_k \nabla^2_{xx} \ell(x_k, y_k) + J_k^T J_k\|_2, \|J_k^T J_k\|_2\}. \tag{3.26}$$

Then, the following hold true:

(i) For some $\kappa_4 \in (0, 1)$, the Cauchy step for subproblem (3.4) yields

$$\Delta q_v(\bar{r}_k; x_k) \geq \kappa_4 \|F_{FEAS}(x_k)\|_2 \min\left\{\theta_k, \frac{1}{1+\Omega}\|F_{FEAS}(x_k)\|_2\right\}. \tag{3.27}$$

(ii) For some $\kappa_5 \in (0, 1)$, the Cauchy step for subproblem (3.8) yields

$$\Delta q(\bar{s}_k; x_k, y_k, \mu_k) \geq \kappa_5 \|F_{AL}(x_k, y_k, \mu_k)\|_2 \min\left\{\Theta_k, \frac{1}{1+\Omega}\|F_{AL}(x_k, y_k, \mu_k)\|_2\right\}. \tag{3.28}$$

Proof We first show (3.27). We know from [35, Theorem 4.4] and (3.10) that

$$\Delta q_v(\bar{r}_k; x_k) \geq \epsilon_r \bar{\kappa}_4 \|F_{\text{FEAS}}(x_k)\|_2 \min\left\{\theta_k, \frac{1}{\Sigma_k}\|F_{\text{FEAS}}(x_k)\|_2\right\}$$

for some $\bar{\kappa}_4 \in (0, 1)$ with $\Sigma_k := 1 + \sup\{|\omega_k(r)| : 0 < \|r\|_2 \leq \theta_k\}$ and

$$\omega_k(r) := \frac{-\Delta q_v(r; x_k) - r^T J_k^T c_k}{\|r\|_2^2}.$$

By rewriting $\omega_k(r)$ and using (3.26), the Cauchy-Schwartz inequality, and standard norm inequalities, we have that

$$\omega_k(r) = \frac{r^T J_k^T J_k r}{2\|r\|_2^2} \leq \Omega.$$

Therefore, $\Sigma_k \leq 1 + \Omega$ and (3.27) follows immediately with $\kappa_4 := \epsilon_r \bar{\kappa}_4$.

Now we show (3.28). We know from [35, Theorem 4.4] and (3.10) that

$$\Delta q(\bar{s}_k; x_k, y_k, \mu_k) \geq \frac{\epsilon_k + \epsilon_r}{2} \bar{\kappa}_5 \|F_{\text{AL}}(x_k, y_k, \mu_k)\|_2 \min\left\{\Theta_k, \frac{1}{\bar{\Sigma}_k}\|F_{\text{AL}}(x_k, y_k, \mu_k)\|_2\right\}$$

for some $\bar{\kappa}_5 \in (0, 1)$ with $\bar{\Sigma}_k := 1 + \sup\{|\bar{\omega}_k(s)| : 0 < \|s\|_2 \leq \Theta_k\}$ and

$$\bar{\omega}_k(s) := \frac{-\Delta q(s; x_k, y_k, \mu_k) - s^T \nabla_x \mathcal{L}(x_k, y_k, \mu_k)}{\|s\|_2^2}.$$

By rewriting $\bar{\omega}_k(s)$ and using (3.26), we have that

$$\begin{aligned}\bar{\omega}_k(s) &= \frac{\mu_k s^T \nabla^2_{xx}\ell(x_k, y_k, \mu_k)s + s^T J_k^T J_k s}{2\|s\|_2^2} \\ &\leq \frac{\|\mu_k \nabla^2_{xx}\ell(x_k, y_k, \mu_k) + J_k^T J_k\|_2 \|s\|_2^2}{2\|s\|_2^2} \leq \Omega.\end{aligned}$$

Therefore, $\bar{\Sigma}_k \leq 1 + \Omega$ and (3.28) follows immediately with $\kappa_5 := \frac{1}{2}\epsilon_r \bar{\kappa}_5 \leq \frac{1}{2}(\varepsilon_k + \varepsilon_r)\bar{\kappa}_5$. □

In the following theorem, we combine the previous lemmas to prove that Algorithm 4 is well-posed.

Theorem 3.6 *The kth iteration of Algorithm 4 is well-posed. That is, either the algorithm will terminate in line 6 or 9, or it will compute $\mu_k > 0$ such that $F_{\text{AL}}(x_k, y_k, \mu_k) \neq 0$ and for the steps $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$ the conditions in (3.11) will be satisfied, in which case $(x_{k+1}, y_{k+1}, \mu_{k+1})$ will be computed.*

Proof If in the kth iteration Algorithm 4 terminates in line 6 or 9, then there is nothing to prove. Therefore, for the remainder of the proof, we assume that line 11 is reached. Lemma 3.2 then ensures that

$$F_{\text{AL}}(x_k, y_k, \mu) \neq 0 \quad \text{for all} \quad \mu > 0 \quad \text{sufficiently small.} \tag{3.29}$$

Consequently, the **while** loop in line 11 will terminate for a sufficiently small $\mu_k > 0$.

By construction, conditions (3.11a) and (3.11b) are satisfied for any $\mu_k > 0$ by $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$. Thus, all that remains is to show that for a sufficiently small $\mu_k > 0$, (3.11c) is also satisfied by $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$. From (3.18d), we have that

$$\lim_{\mu \to 0} \Delta q_v(s_k; x_k) = \lim_{\mu \to 0} \Delta q_v(\bar{s}_k; x_k) = \Delta q_v(\bar{r}_k; x_k) = \Delta q_v(r_k; x_k). \tag{3.30}$$

If $\Delta q_v(r_k; x_k) > 0$, then (3.30) implies that (3.11c) will be satisfied for sufficiently small $\mu_k > 0$. On the other hand, suppose

$$\Delta q_v(r_k; x_k) = \Delta q_v(\bar{r}_k; x_k) = 0, \tag{3.31}$$

which along with (3.27) and the definitions of θ_k and $\delta_k > 0$, must mean that $F_{\text{FEAS}}(x_k) = 0$. If $c_k \neq 0$, then Algorithm 4 would have terminated in line 9 and, therefore, we must have $c_k = 0$. This and (3.31) imply that

$$\min\{\kappa_3 \Delta q_v(r_k; x_k), v_k - \tfrac{1}{2}(\kappa_t t_j)^2\} = -\tfrac{1}{2}(\kappa_t t_j)^2 < 0 \tag{3.32}$$

since $t_j > 0$ by construction and $\kappa_t \in (0, 1)$ by choice. Therefore, we can deduce that (3.11c) will be satisfied for sufficiently small $\mu_k > 0$ by observing (3.30), (3.31) and (3.32). Combining this with (3.29) and the fact that the **while** loop on line 19

ensures that μ_k will eventually be as small as required, guarantees that the **while** loop will terminate finitely. This completes the proof as all remaining steps in the kth iteration are well-posed and explicit. □

3.2 Global convergence

We analyze the global convergence properties of Algorithm 4 under the assumption that the algorithm does not terminate finitely. That is, in this section we assume that neither a first-order optimal solution nor an infeasible stationary point is found after a finite number of iterations so that the sequence $\{(x_k, y_k, \mu_k)\}_{k\geq 0}$ is infinite.

We provide global convergence guarantees under the following assumption. This assumption is assumed throughout this section and is therefore not stated explicitly in each result.

Assumption 3.2 *The iterates $\{x_k\}_{k\geq 0}$ are contained in a convex compact set over which the objective function f and constraint function c are both twice-continuously differentiable.*

This assumption and the bound on the multipliers enforced in line 37 of Algorithm 4 imply that there exists a positive monotonically increasing sequence $\{\Omega_j\}_{j\geq 1}$ such that for all $k_j \leq k < k_{j+1}$ we have

$$\|\nabla^2_{xx}\mathcal{L}(\sigma, y_k, \mu_k)\|_2 \leq \Omega_j \quad \text{for all} \quad \sigma \quad \text{on the segment} \quad [x_k, x_k+s_k], \tag{3.33a}$$

$$\|\mu_k \nabla^2_{xx}\ell(x_k, y_k) + J_k^T J_k\|_2 \leq \Omega_j, \tag{3.33b}$$

$$\text{and} \quad \|J_k^T J_k\|_2 \leq \Omega_j. \tag{3.33c}$$

We begin our analysis in this section by proving the following lemma, which provides critical bounds on differences in (components of) the augmented Lagrangian summed over sequences of iterations.

Lemma 3.7 *The following hold true.*

(i) If $\mu_k = \mu$ for some $\mu > 0$ and all sufficiently large k, then there exist positive constants M_f, M_c, and $M_{\mathcal{L}}$ such that for all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1} \mu_k (f_k - f_{k+1}) < M_f, \tag{3.34}$$

$$\sum_{k=0}^{p-1} \mu_k y_k^T (c_{k+1} - c_k) < M_c, \tag{3.35}$$

$$\text{and} \quad \sum_{k=0}^{p-1} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) < M_{\mathcal{L}}. \tag{3.36}$$

(ii) If $\mu_k \to 0$, then the sums

$$\sum_{k=0}^{\infty} \mu_k (f_k - f_{k+1}), \tag{3.37}$$

$$\sum_{k=0}^{\infty} \mu_k y_k^T (c_{k+1} - c_k), \tag{3.38}$$

$$\text{and} \quad \sum_{k=0}^{\infty} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) \tag{3.39}$$

converge and are finite, and

$$\lim_{k \to \infty} \|c_k\|_2 = \bar{c} \ \text{ for some } \ \bar{c} \geq 0. \tag{3.40}$$

Proof Under Assumption 3.2 we may conclude that for some constant $\overline{M}_f > 0$ and all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1} (f_k - f_{k+1}) = f_0 - f_p < \overline{M}_f.$$

If $\mu_k = \mu$ for all sufficiently large k, then this implies that (3.34) clearly holds for some sufficiently large M_f. Otherwise, if $\mu_k \to 0$, then it follows from Dirichlet's Test [12, §3.4.10] and the fact that $\{\mu_k\}_{k \geq 0}$ is a monotonically decreasing sequence that converges to zero that (3.37) converges and is finite.

Next, we show that for some constant $\overline{M}_c > 0$ and all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1} y_k^T (c_{k+1} - c_k) < \overline{M}_c. \tag{3.41}$$

First, suppose that $\mathcal{Y}$ defined in (3.16) is finite. It follows that there exists $k' \geq 0$ and y such that $y_k = y$ for all $k \geq k'$. Moreover, under Assumption 3.2 there exists a constant $\widehat{M}_c > 0$ such that for all $p \geq k' + 1$ we have

$$\sum_{k=k'}^{p-1} y_k^T (c_{k+1} - c_k) = y^T \sum_{k=k'}^{p-1} (c_{k+1} - c_k) = y^T (c_p - c_{k'}) \leq \|y\|_2 \|c_p - c_{k'}\|_2 < \widehat{M}_c.$$

It is now clear that (3.41) holds in this case. Second, suppose that $|\mathcal{Y}| = \infty$ so that the sequence $\{k_j\}_{j \geq 1}$ in Algorithm 4 is infinite. By construction $t_j \to 0$, so for some $j' \geq 1$ we have

$$t_j = t_{j-1}^{1+\epsilon} \quad \text{and} \quad Y_{j+1} = t_{j-1}^{-\epsilon} \ \text{ for all } \ j \geq j'. \tag{3.42}$$

From the definition of the sequence $\{k_j\}_{j\geq 1}$, (3.14), and (3.15), we know that

$$\sum_{k=k_j}^{k_{j+1}-1} y_k^T(c_{k+1}-c_k) = y_{k_j}^T \sum_{k=k_j}^{k_{j+1}-1} (c_{k+1}-c_k) = y_{k_j}^T(c_{k_{j+1}}-c_{k_j})$$
$$\leq \|y_{k_j}\|_2 \|c_{k_{j+1}}-c_{k_j}\|_2 \leq 2Y_{j+1}t_j = 2t_{j-1} \quad \text{for all} \quad j \geq j',$$

where the last equality follows from (3.42). Using these relationships, summing over all $j' \leq j \leq j'+q$ for an arbitrary integer $q \geq 1$, and using the fact that $t_{j+1} \leq \gamma_t t_j$ by construction, leads to

$$\sum_{j=j'}^{j'+q} \left(\sum_{k=k_j}^{k_{j+1}-1} y_k^T(c_{k+1}-c_k) \right) \leq 2 \sum_{j=j'}^{j'+q} t_{j-1}$$
$$\leq 2t_{j'-1} \sum_{l=0}^{q} \gamma_t^l = 2t_{j'-1} \frac{1-\gamma_t^{q+1}}{1-\gamma_t} \leq \frac{2t_{j'-1}}{1-\gamma_t}.$$

It is now clear that (3.41) holds in this case as well.

We have shown that (3.41) always holds. Thus, if $\mu_k = \mu$ for all sufficiently large k, then (3.35) holds for some sufficiently large M_c. Otherwise, if $\mu_k \to 0$, then it follows from Dirichlet's Test [12, §3.4.10], (3.41) and the fact that $\{\mu_k\}_{k\geq 0}$ is a monotonically decreasing sequence that converges to zero that (3.38) converges and is finite.

Finally, observe that

$$\sum_{k=0}^{p-1} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)$$
$$= \sum_{k=0}^{p-1} \mu_k(f_k - f_{k+1}) + \sum_{k=0}^{p-1} \mu_k y_k^T(c_{k+1}-c_k) + \tfrac{1}{2}\sum_{k=0}^{p-1}(\|c_k\|_2^2 - \|c_{k+1}\|_2^2)$$
$$= \sum_{k=0}^{p-1} \mu_k(f_k - f_{k+1}) + \sum_{k=0}^{p-1} \mu_k y_k^T(c_{k+1}-c_k) + \tfrac{1}{2}(\|c_0\|_2^2 - \|c_p\|_2^2). \tag{3.43}$$

If $\mu_k = \mu$ for all sufficiently large k, then it follows from Assumption 3.2, (3.34), (3.35), and (3.43) that (3.36) will hold for some sufficiently large $M_{\mathcal{L}}$. Otherwise, consider when $\mu_k \to 0$. Taking the limit of (3.43) as $p \to \infty$, we have from Assumption 3.2 and conditions (3.37) and (3.38) that

$$\sum_{k=0}^{\infty} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) < \infty.$$

Since the terms in this sum are all nonnegative, it follows from the Monotone Convergence Theorem that (3.39) converges and is finite. Moreover, we may again take the

limit of (3.43) as $p \to \infty$ and use (3.37), (3.38), and (3.39) to conclude that (3.40) holds. □

In the following subsections, we consider different situations depending on the number of times that the Lagrange multiplier vector is updated.

3.2.1 Finite number of multiplier updates

In this section, we consider the case when $\mathcal{Y}$ in (3.16) is finite. In this case, the counter j in Algorithm 4, which tracks the number of times that the dual vector is updated, satisfies

$$j \in \{1, 2, \ldots \bar{j}\} \text{ for some finite } \bar{j}. \tag{3.44}$$

For the purposes of our analysis in this section, we define

$$t := t_{\bar{j}} > 0 \text{ and } T := T_{\bar{j}} > 0. \tag{3.45}$$

We consider two subcases depending on whether the penalty parameter stays bounded away from zero, or if it converges to zero. First we consider cases when it converges to zero.

Lemma 3.8 *If $|\mathcal{Y}| < \infty$ and $\mu_k \to 0$, then there exist a vector y and integer $\bar{k} \geq 0$ such that*

$$y_k = y \text{ for all } k \geq \bar{k}, \tag{3.46}$$

and for some constant $\bar{c} > 0$, we have the limits

$$\lim_{k\to\infty} \|c_k\|_2 = \bar{c} > 0 \text{ and } \lim_{k\to\infty} F_{\text{FEAS}}(x_k) = 0. \tag{3.47}$$

Therefore, every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point.

Proof Since $|\mathcal{Y}| < \infty$, we know that (3.44) and (3.45) both hold for some $\bar{j} \geq 0$. It follows by construction in Algorithm 4 that there exists y and a scalar $\bar{k} \geq k_{\bar{j}}$ such that (3.46) holds.

From (3.40), it follows that $\|c_k\|_2 \to \bar{c}$ for some $\bar{c} \geq 0$. If $\bar{c} = 0$, then by Assumption 3.2, the definition of $\nabla_x \mathcal{L}$, (3.46), and the fact that $\mu_k \to 0$ it follows that $\lim_{\mu\to 0} \nabla_x \mathcal{L}(x_k, y, \mu_k) = J_k^T c_k = 0$, which implies that $\lim_{\mu\to 0} F_{\text{AL}}(x_k, y, \mu_k) = F_{\text{FEAS}}(x_k) = 0$. This would imply that for some $k \geq \bar{k}$ the algorithm would set $j \leftarrow \bar{j} + 1$, violating (3.44). Thus, we conclude that $\bar{c} > 0$, which proves the first part of (3.47).

Next, we prove that

$$\liminf_{k\geq 0} F_{\text{FEAS}}(x_k) = 0. \tag{3.48}$$

If (3.48) does not hold, then there exists $\zeta \in (0, 1)$ and $k' \geq \bar{k}$ such that

$$\|F_{\text{FEAS}}(x_k)\|_2 \geq 2\zeta \text{ for all } k \geq k'. \tag{3.49}$$

Hence, by (3.49) and the fact that $\mu_k \to 0$, there exists $k'' \geq k'$ such that

$$\|F_{\text{AL}}(x_k, y, \mu_k)\|_2 \geq \zeta \quad \text{for all} \quad k \geq k''. \tag{3.50}$$

We now show that the trust region radius Θ_k is bounded away from zero for $k \geq k''$. In order to see this, suppose that for some $k \geq k''$ we have

$$0 < \Theta_k \leq \min\left\{ \frac{\zeta}{1 + \Omega_{\bar{j}}}, \frac{(1 - \eta_{vs})\kappa_1\kappa_5\zeta}{1 + \Omega_{\bar{j}}} \right\} =: \delta_{\text{thresh}} \tag{3.51}$$

where $\{\Omega_j\}_{j \geq 1}$ is defined with (3.33). It then follows from (3.11a), (3.28), (3.33b), (3.50), and (3.51) that

$$\Delta q(s_k; x_k, y, \mu_k) \geq \kappa_1 \Delta q(\bar{s}_k; x_k, y, \mu_k) \geq \kappa_1\kappa_5\zeta \min\left\{ \frac{\zeta}{1 + \Omega_{\bar{j}}}, \Theta_k \right\} \geq \kappa_1\kappa_5\zeta\Theta_k. \tag{3.52}$$

Using the definition of q, Taylor's Theorem, (3.33a), (3.33b), and the trust-region constraint, we may conclude that for some σ_k on the segment $[x_k, x_k + s_k]$ we have

$$\begin{aligned} |q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)| &= \tfrac{1}{2}\left| s_k^T(\mu_k \nabla^2_{xx}\ell(x_k, y_k) + J_k^T J_k)s_k \right. \\ &\quad \left. - s_k^T \nabla^2_{xx}\mathcal{L}(\sigma_k, y, \mu_k)s_k \right| \\ &\leq \Omega_{\bar{j}}\|s_k\|_2^2 \leq \Omega_{\bar{j}}\Theta_k^2. \end{aligned} \tag{3.53}$$

The definition of ρ_k, (3.51), (3.52), and (3.53) then yield

$$|\rho_k - 1| = \left| \frac{q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)}{\Delta q(s_k; x_k, y, \mu_k)} \right| \leq \frac{\Omega_{\bar{j}}\Theta_k^2}{\kappa_1\kappa_5\zeta\Theta_k} = \frac{\Omega_{\bar{j}}\Theta_k}{\kappa_1\kappa_5\zeta} \leq 1 - \eta_{vs}.$$

This implies that the **if** clause in line 26 of Algorithm 4 will be true, and along with (3.50) we may conclude that the trust region radius will not be decreased any further. Consequently, we have shown that the trust region radius updating strategy in Algorithm 4 guarantees that for some $\delta_{\text{min}} \in (0, \delta_{\text{thresh}})$ we have

$$\Theta_k \geq \delta_{\text{min}} \quad \text{for all} \quad k \geq k''. \tag{3.54}$$

Now, since Θ_k is bounded below, there must exist an infinite subsequence, indexed by an ordered set $\mathcal{S} \subseteq \mathbb{N}$, of successful iterates. If we define $\mathcal{S}'' := \{k \in \mathcal{S} : k \geq k''\}$, then we may conclude from the fact that $x_{k+1} = x_k$ when $k \notin \mathcal{S}$, (3.46), (3.12), (3.52), and (3.54) that

$$\begin{aligned}\sum_{k=k''}^{\infty} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) &= \sum_{k\in\mathcal{S}''} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \\ &\geq \sum_{k\in\mathcal{S}''} \eta_s \Delta q(s_k; x_k, y, \mu_k) \geq \sum_{k\in\mathcal{S}''} \eta_s \kappa_1 \kappa_5 \zeta \delta_{\min} \\ &= \infty, \end{aligned} \tag{3.55}$$

contradicting (3.39). Therefore, we conclude that (3.48) holds.

Now we prove the second part of (3.47). By contradiction, suppose $F_{\text{FEAS}}(x_k) \nrightarrow 0$. This supposition and (3.48) imply that the subsequence of successful iterates indexed by $\mathcal{S}$ is infinite and there exists a constant $\varepsilon \in (0, 1)$ and sequences $\{\underline{k}_i\}_{i\geq 0}$ and $\{\overline{k}_i\}_{i\geq 0}$ defined in the following manner: $\underline{k}_0$ is the first iterate in $\mathcal{S}$ such that $\|F_{\text{FEAS}}(x_{\underline{k}_0})\|_2 \geq 4\varepsilon > 0$; $\overline{k}_i$ for $i \geq 0$ is the first iterate strictly greater than $\underline{k}_i$ such that

$$\|F_{\text{FEAS}}(x_{\overline{k}_i})\|_2 < 2\varepsilon; \tag{3.56}$$

and $\underline{k}_i$ for $i \geq 1$ is the first iterate in $\mathcal{S}$ strictly greater than $\overline{k}_{i-1}$ such that

$$\|F_{\text{FEAS}}(x_{\underline{k}_i})\|_2 \geq 4\varepsilon > 0. \tag{3.57}$$

We may now define

$$\mathcal{K} := \{k \in \mathcal{S} : \underline{k}_i \leq k < \overline{k}_i \ \text{ for some } \ i \geq 0\}.$$

Since $\mu_k \to 0$, we may use (3.46) to conclude that there exists k''' such that

$$\|F_{\text{AL}}(x_k, y, \mu_k)\|_2 \geq \varepsilon \ \text{ for all } \ k \in \mathcal{K} \ \text{ such that } \ k \geq k'''. \tag{3.58}$$

It follows from the definition of $\mathcal{K}$, (3.11a), (3.28), (3.46), (3.33b), and (3.58) that

$$\begin{aligned}\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) &\geq \eta_s \Delta q(s_k; x_k, y_k, \mu_k) \\ &\geq \eta_s \kappa_1 \kappa_5 \varepsilon \min\left\{\frac{\varepsilon}{1+\Omega_{\bar{j}}}, \Theta_k\right\} \ \text{ for all } \ k \in \mathcal{K} \ \text{ such that } \ k \geq k'''. \end{aligned} \tag{3.59}$$

It also follows from (3.39) and since $\mathcal{L}(x_{k+1}, y_k, \mu_k) \leq \mathcal{L}(x_k, y_k, \mu_k)$ for all $k \geq 0$ that

$$\begin{aligned}\infty &> \sum_{k=0}^{\infty} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \\ &= \sum_{k\in\mathcal{S}} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \geq \sum_{k\in\mathcal{K}} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k). \end{aligned} \tag{3.60}$$

Summing (3.59) for $k \in \mathcal{K}$ and using (3.60) yields $\lim_{k\in\mathcal{K}} \Theta_k = 0$, which combined with (3.59) and (3.46) leads to

$$\mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \geq \eta_s \kappa_1 \kappa_5 \varepsilon \Theta_k > 0 \quad \text{for all sufficiently large} \quad k \in \mathcal{K}. \tag{3.61}$$

By the triangle-inequality and (3.61), there exists some $\bar{i} \geq 1$ such that

$$\begin{aligned} \|x_{\underline{k}_i} - x_{\bar{k}_i}\|_2 &\leq \sum_{j=\underline{k}_i}^{\bar{k}_i - 1} \|x_j - x_{j+1}\|_2 = \sum_{j=\underline{k}_i}^{\bar{k}_i - 1} \|s_j\|_2 \leq \sum_{j=\underline{k}_i}^{\bar{k}_i - 1} \Theta_j \\ &\leq \frac{1}{\eta_s \kappa_1 \kappa_5 \varepsilon} \sum_{j=\underline{k}_i}^{\bar{k}_i - 1} \mathcal{L}(x_j, y, \mu_j) - \mathcal{L}(x_{j+1}, y, \mu_j) \quad \text{for} \quad i \geq \bar{i}. \end{aligned}$$

Summing over all $i \geq \bar{i}$ and using (3.39), we find

$$\begin{aligned} \sum_{i=\bar{i}}^{\infty} \|x_{\underline{k}_i} - x_{\bar{k}_i}\|_2 &\leq \frac{1}{\eta_s \kappa_1 \kappa_5 \varepsilon} \sum_{i=\bar{i}}^{\infty} \left(\sum_{j=\underline{k}_i}^{\bar{k}_i - 1} \mathcal{L}(x_j, y, \mu_j) - \mathcal{L}(x_{j+1}, y, \mu_j) \right) \\ &\leq \frac{1}{\eta_s \kappa_1 \kappa_5 \varepsilon} \sum_{k\in\mathcal{K}} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \\ &\leq \frac{1}{\eta_s \kappa_1 \kappa_5 \varepsilon} \sum_{k=0}^{\infty} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) < \infty, \end{aligned}$$

which implies that

$$\lim_{i\to\infty} \|x_{\underline{k}_i} - x_{\bar{k}_i}\|_2 = 0.$$

It follows from the previous limit, (3.57), and Assumption 3.2 that for i sufficiently large we have $\|F_{\text{FEAS}}(x_{\bar{k}_i})\|_2 > 2\varepsilon$, contradicting (3.56). We may conclude that the second part of (3.47) holds.

The fact that every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point follows from (3.47). □

The next lemma considers the case when μ stays bounded away from zero. This is possible, for example, if the algorithm converges to an infeasible stationary point that is stationary for the augmented Lagrangian.

Lemma 3.9 *If $|\mathcal{Y}| < \infty$ and $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, then with t defined in (3.45) there exist a vector y and integer $\bar{k} \geq 0$ such that*

$$y_k = y \quad \text{and} \quad \|c_k\|_2 \geq t \quad \text{for all} \quad k \geq \bar{k}, \tag{3.62}$$

and we have the limit

$$\lim_{k\to\infty} F_{FEAS}(x_k) = 0. \tag{3.63}$$

Therefore, every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point.

Proof Since $|\mathcal{Y}| < \infty$, we know that (3.44) and (3.45) both hold for some $\bar{j} \geq 0$. Since we also suppose that $\mu_k = \mu > 0$ for all sufficiently large k, it follows by construction in Algorithm 4 that there exists y and a scalar $k' \geq k_{\bar{j}}$ such that

$$\mu_k = \mu \quad \text{and} \quad y_k = y \quad \text{for all} \quad k \geq k'. \tag{3.64}$$

Next, we prove that

$$\liminf_{k \geq 0} \|F_{\text{AL}}(x_k, y, \mu)\|_2 = 0. \tag{3.65}$$

If (3.65) does not hold, then there exists $\zeta \in (0, 1)$ and $k'' \geq k'$ such that

$$\|F_{\text{AL}}(x_k, y, \mu)\|_2 \geq \zeta \quad \text{for all} \quad k \geq k''. \tag{3.66}$$

We now show that the trust region radius Θ_k is bounded away from zero for $k \geq k''$. In order to see this, suppose that for some $k \geq k''$ we have

$$0 < \Theta_k \leq \min \left\{ \frac{\zeta}{1 + \Omega_{\bar{j}}}, \frac{(1 - \eta_{vs})\kappa_1 \kappa_5 \zeta}{1 + \Omega_{\bar{j}}} \right\} =: \delta_{\text{thresh}} \tag{3.67}$$

where $\{\Omega_j\}_{j \geq 1}$ is defined with (3.33). It then follows from (3.11a), (3.28), (3.33b), (3.66), and (3.67) that

$$\Delta q(s_k; x_k, y, \mu) \geq \kappa_1 \Delta q(\bar{s}_k; x_k, y, \mu) \geq \kappa_1 \kappa_5 \zeta \min \left\{ \frac{\zeta}{1 + \Omega_{\bar{j}}}, \Theta_k \right\} \geq \kappa_1 \kappa_5 \zeta \Theta_k. \tag{3.68}$$

Using the definition of q, Taylor's Theorem, (3.33a), (3.33b), and the trust-region constraint, we may conclude that for some σ_k on the segment $[x_k, x_k + s_k]$ we have

$$\begin{aligned} &|q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)| \\ &= \tfrac{1}{2} \left| s_k^T (\mu_k \nabla_{xx}^2 \ell(x_k, y_k) + J_k^T J_k) s_k - s_k^T \nabla_{xx}^2 \mathcal{L}(\sigma_k, y, \mu_k) s_k \right| \\ &\leq \Omega_{\bar{j}} \|s_k\|_2^2 \leq \Omega_{\bar{j}} \Theta_k^2. \end{aligned} \tag{3.69}$$

The definition of ρ_k, (3.69), (3.68), and (3.67) then yield

$$\begin{aligned} |\rho_k - 1| &= \left| \frac{q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)}{\Delta q(s_k; x_k, y, \mu_k)} \right| \\ &\leq \frac{\Omega_{\bar{j}} \Theta_k^2}{\kappa_1 \kappa_5 \zeta \Theta_k} = \frac{\Omega_{\bar{j}} \Theta_k}{\kappa_1 \kappa_5 \zeta} \leq 1 - \eta_{vs}. \end{aligned}$$

This implies that a very successful iteration will occur and along with the trust region radius updating strategy in Algorithm 4, we may conclude that for some $\delta_{\min} \in (0, \delta_{\text{thresh}})$ we have

$$\Theta_k \geq \delta_{\min} \quad \text{for all} \quad k \geq k''. \tag{3.70}$$

Now, since Θ_k is bounded below, there must exist an infinite subsequence, indexed by an ordered set $\mathcal{S} \subseteq \mathbb{N}$, of successful iterates. If we define $\mathcal{S}'' := \{k \in \mathcal{S} : k \geq k''\}$, then we may conclude from the fact that $x_{k+1} = x_k$ when $k \notin \mathcal{S}$, (3.64), (3.12), (3.68), and (3.70) that

$$\sum_{k=k''}^{\infty} \mathcal{L}(x_k, y, \mu) - \mathcal{L}(x_{k+1}, y, \mu) = \sum_{k \in \mathcal{S}''} \mathcal{L}(x_k, y, \mu) - \mathcal{L}(x_{k+1}, y, \mu)$$
$$\geq \sum_{k \in \mathcal{S}''} \eta_s \Delta q(s_k; x_k, y, \mu) \geq \sum_{k \in \mathcal{S}''} \eta_s \kappa_1 \kappa_5 \zeta \delta_{\min} = \infty,$$

contradicting (3.36). Therefore, we conclude that (3.65) holds. Moreover, the same argument used in the second part of the proof of Lemma 3.8 (with F_{FEAS} replaced by F_{AL}) shows that

$$\lim_{k \to \infty} \|F_{\text{AL}}(x_k, y, \mu)\|_2 = 0. \tag{3.71}$$

It then follows that there exists $\bar{k} \geq k'$ such that $\|c_k\|_2 \geq t$ for all $k \geq \bar{k}$, since otherwise it follows from (3.71) that for some $k \geq \bar{k}$ Algorithm 4 sets $j \leftarrow \bar{j} + 1$, violating (3.44). Thus, we have shown that (3.62) holds.

Next, we turn to the limits in (3.63). It follows from (3.71) and part (i) of Lemma 3.3 that

$$\lim_{k \to \infty} \|s_k\|_2 \leq \lim_{k \to \infty} \Theta_k = \lim_{k \to \infty} \Gamma_k \min\{\delta_k, \delta \|P[x_k - \nabla_x \mathcal{L}(x_k, y, \mu)] - x_k\|_2\} = 0. \tag{3.72}$$

From (3.72) and Assumption 3.2, we have

$$\lim_{k \to \infty} \Delta q_v(s_k; x_k) = 0, \tag{3.73}$$

and from the definition of v, (3.45), and (3.62) that

$$v_k - \tfrac{1}{2}(\kappa_t t_{\bar{j}})^2 \geq \tfrac{1}{2} t^2 - \tfrac{1}{2}(\kappa_t t)^2 = \tfrac{1}{2}(1 - \kappa_t^2) t^2 > 0 \quad \text{for all} \quad k \geq \bar{k}. \tag{3.74}$$

We now prove that $F_{\text{FEAS}}(x_k) \to 0$. To see this, first note that

$$F_{\text{AL}}(x_k, y, \mu) \neq 0 \quad \text{for all} \quad k \geq \bar{k},$$

or else the algorithm would set $\mu_{k+1} < \mu$ in line 12 of Algorithm 4, violating (3.64). Standard trust-region theory [11] then ensures that there will be infinitely many successful iterations, which we denote by $\mathcal{S}$, for $k \geq \bar{k}$. If we suppose that $F_{\text{FEAS}}(x_k) \not\to 0$, then for some $\zeta \in (0, 1)$ there exists an infinite subsequence indexed by

$$\mathcal{S}_\zeta := \{k \in \mathcal{S} : k \geq \bar{k} \quad \text{and} \quad \|F_{\text{FEAS}}(x_{k+1})\|_2 \geq \zeta\}.$$

We may then observe from the updating strategies for θ_k and δ_k that

$$\begin{aligned}\theta_{k+1} &= \min\{\delta_{k+1}, \delta\|F_{\text{FEAS}}(x_{k+1})\|_2\} \\ &\geq \min\{\max\{\delta_R, \delta_k\}, \delta\zeta\} \geq \min\{\delta_R, \delta\zeta\} > 0 \quad \text{for all} \quad k \in \mathcal{S}_\zeta. \end{aligned} \tag{3.75}$$

Using (3.11b), (3.27), (3.33c), (3.44), and (3.75), we then find for $k \in \mathcal{S}_\zeta$ that

$$\Delta q_v(r_{k+1}; x_{k+1}) \geq \kappa_2 \Delta q_v(\bar{r}_{k+1}; x_{k+1}) \geq \kappa_2\kappa_4\zeta \min\left\{\frac{\zeta}{1+\Omega_{\bar{j}}}, \delta_R, \delta\zeta\right\} =: \zeta' > 0.$$

We may now combine the previous equation, (3.74), and (3.73) to state that (3.11c) must be violated for sufficiently large $k \in \mathcal{S}_\zeta$ and, consequently, the penalty parameter will be decreased. However, this is a contradiction to (3.64), so we conclude that $F_{\text{FEAS}}(x_k) \to 0$. The fact that every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point follows since $\|c_k\|_2 \geq t$ for all $k \geq \bar{k}$ from (3.62) and $F_{\text{FEAS}}(x_k) \to 0$. □

This completes the analysis for the case that the set $\mathcal{Y}$ is finite. The next section considers the complementarity situation when the Lagrange multiplier vector is updated an infinite number of times.

3.2.2 Infinite number of multiplier updates

We now consider the case when $|\mathcal{Y}| = \infty$. In this case, it follows by construction in Algorithm 4 that

$$\lim_{j\to\infty} t_j = \lim_{j\to\infty} T_j = 0. \tag{3.76}$$

As in the previous subsection, we consider two subcases depending on whether the penalty parameter remains bounded away from zero, or if it converges to zero. Our next lemma shows that when the penalty parameter does remain bounded away from zero, then a subsequence of iterates converges to a first-order optimal point. In general, this is the ideal case for a feasible problem.

Lemma 3.10 *If $|\mathcal{Y}| = \infty$ and $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, then*

$$\lim_{j\to\infty} c_{k_j} = 0 \tag{3.77}$$

$$\text{and} \quad \lim_{j\to\infty} F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j}) = 0. \tag{3.78}$$

Thus, any limit point (x_*, y_*) of $\{(x_{k_j}, \widehat{y}_{k_j})\}_{j\geq 0}$ is first-order optimal for (2.1).

Proof Since $|\mathcal{Y}| = \infty$, it follows that the condition in line 32 holds an infinite number of times. The limit (3.77) then follows by (3.76) since line 35 sets $k_j \leftarrow k + 1$ for all $k_j \in \mathcal{Y}$.

To prove (3.78), we first define

$$\mathcal{Y}' = \{k_j \in \mathcal{Y} : \|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2 \leq \|F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})\|_2\}.$$

It follows from (3.76) and line 34 of Algorithm 4 that if $\mathcal{Y}'$ is infinite, then

$$\lim_{k_j \in \mathcal{Y}'} F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j}) = 0. \tag{3.79}$$

Meanwhile, it follows from (3.76) and line 34 of Algorithm 4 that if $\mathcal{Y}\backslash\mathcal{Y}'$ is infinite, then

$$\lim_{k_j \in \mathcal{Y}\backslash\mathcal{Y}'} F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1}) = 0. \tag{3.80}$$

Under Assumption 3.2, (3.80) may be combined with (3.77) and the fact that $\mu_k = \mu$ for some $\mu > 0$ to deduce that if $\mathcal{Y}\backslash\mathcal{Y}'$ is infinite, then

$$\lim_{k_j \in \mathcal{Y}\backslash\mathcal{Y}'} F_{\text{L}}(x_{k_j}, y_{k_j-1}) = 0. \tag{3.81}$$

We may now combine (3.81) with (3.13) to state that if $\mathcal{Y}\backslash\mathcal{Y}'$ is infinite, then

$$\lim_{k_j \in \mathcal{Y}\backslash\mathcal{Y}'} F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j}) = 0. \tag{3.82}$$

The desired result (3.78) now follows from (3.79), (3.82), and the supposition that $|\mathcal{Y}| = \infty$. □

We now prove a corollary showing that if Algorithm 4 employs a particular update for $\widehat{y}_{k+1}$ in line 33 (satisfying (3.13)), then a subsequence of multiplier estimates $\{\widehat{y}_k\}$ converges to an optimal Lagrange multiplier vector when the linear independence constraint qualification (LICQ) holds at limit points. For this result only, we make the following additional assumption.

Assumption 3.3 *If x_* is a limit point of $\{x_k\}$ that is feasible for problem (2.1), then the index set $\mathcal{I}(x_*) := \{i : [x_*]_i > 0\}$ is nonempty and the matrix $J_{\mathcal{I}}(x_*)$ that contains the subset of columns of $J(x_*)$ corresponding to the index set $\mathcal{I}(x_*)$ has full row rank.*

Corollary 3.11 *If $|\mathcal{Y}| = \infty$, $\mu_k = \mu > 0$ for all sufficiently large k, and line 33 of Algorithm 4 sets*

$$\widehat{y}_{k+1} \leftarrow \begin{cases} y_k & \text{if } \|F_L(x_{k+1}, y_k)\|_2 \leq \|F_L(x_{k+1}, \pi(x_{k+1}, y_k, \mu_k))\|_2, \\ \pi(x_{k+1}, y_k, \mu_k) & \text{otherwise,} \end{cases} \tag{3.83}$$

then there exists an infinite ordered set $\mathcal{J} \subseteq \mathbb{N}$ such that

$$\lim_{j \in \mathcal{J}} (x_{k_j}, y_{k_j}) = (x_*, y_*),$$

where (x_*, y_*) a first-order optimal point for problem (2.1), the vector y_* is the (unique) solution of

$$\min_{y \in \mathbb{R}^m} \|g_{\mathcal{I}}(x_*) - J_{\mathcal{I}}(x_*)^T y\|_2^2, \tag{3.84}$$

and $g_{\mathcal{I}}(x_*)$ *and* $\mathcal{J}_{\mathcal{I}}(x_*)^T$ *contain the rows of* $g(x_*)$ *and* $\mathcal{J}(x_*)^T$*, respectively, corresponding to* $\mathcal{I}(x_*)$.

Proof We know from $|\mathcal{Y}| = \infty$ and Assumption 3.2 that there exists an infinite ordered set $\mathcal{J}_1 \subseteq \mathbb{N}$ and a vector x_* such that $\lim_{j \in \mathcal{J}_1} x_{k_j} = x_*$. It also follows from this fact, the assumptions of this corollary, and Lemma 3.10 that

$$\lim_{j \in \mathcal{J}_1} c(x_{k_j}) = 0, \quad \lim_{j \in \mathcal{J}_1} F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j}) = 0, \tag{3.85}$$

and any limit points of $(x_{k_j}, \widehat{y}_{k_j})$ are first-order KKT points for (2.1). Let us define the set

$$\mathcal{J}_2 := \{j \in \mathcal{J}_1 : \|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2 \leq \|F_{\text{AL}}(x_{k_j}, \pi(x_{k_j}, y_{k_j-1}, \mu_{k_j-1}))\|_2\},$$

which is motivated by line 34 of Algorithm 4. Also, we note that

$$\lim_{j \in \mathcal{J}_1} Y_j = \infty \tag{3.86}$$

as a result of (3.76) and line 35 of Algorithm 4. We now consider two cases.

Case 1: Suppose that $|\mathcal{J}_2| = \infty$. It follows from (3.76), line 34 of Algorithm 4, and definition of $\mathcal{J}_2$ that

$$0 = \lim_{j \in \mathcal{J}_2} \|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2 = \lim_{j \in \mathcal{J}_2} \|P\big[x_{k_j} - \big(g(x_{k_j}) - J(x_{k_j})^T \widehat{y}_{k_j}\big)\big] - x_{k_j}\|_2.$$

Using this limit and the definition of $\mathcal{I}(x_*)$, it follows that

$$\lim_{j \in \mathcal{J}_2} \big(g_{\mathcal{I}}(x_{k_j}) - J_{\mathcal{I}}(x_{k_j})^T \widehat{y}_{k_j}\big) = 0,$$

which, when combined with $\lim_{j \in \mathcal{J}_1} x_{k_j} = x_*$, Assumption 3.3, and (3.84), implies that

$$\lim_{j \in \mathcal{J}_2} \widehat{y}_{k_j} = y_* \tag{3.87}$$

so that $(x_*, y_*) = \lim_{j \in \mathcal{J}_2} (x_{k_j}, \widehat{y}_{k_j})$ is a first-order point for (2.1). Using (3.87), the fact that the upper bound on the multipliers increases to infinity in (3.86), and the definition of y_{k+1} in line 37, we then have

$$\lim_{j \in \mathcal{J}_2} y_{k_j} = y_*.$$

Defining $\mathcal{J} := \mathcal{J}_2 \subseteq \mathcal{J}_1$, this completes the proof for this case.

Case 2: Suppose that $|\mathcal{J}_2| < \infty$. Since $|\mathcal{J}_2| < \infty$, it follows from (3.76), line 34 of Algorithm 4, and the fact that $\mu_k = \mu > 0$ for all sufficiently large k that

$$
\begin{aligned}
0 &= \lim_{j \in \mathcal{J}_1} \| F_{\text{AL}}(x_{k_j}, \pi(x_{k_j}, y_{k_j-1}, \mu), \mu) \|_2 \\
&= \lim_{j \in \mathcal{J}_1} \| P\big[x_{k_j} - \mu\big(g(x_{k_j}) - J(x_{k_j})^T \pi(x_{k_j}, y_{k_j-1}, \mu)\big)\big] - x_{k_j} \|_2.
\end{aligned}
$$

Using this limit, the definition of $\mathcal{I}$, and $\lim_{j \in \mathcal{J}_1} x_{k_j} = x_*$ shows that

$$
\lim_{j \in \mathcal{J}_1} g_{\mathcal{I}}(x_{k_j}) - J_{\mathcal{I}}(x_{k_j})^T \pi(x_{k_j}, y_{k_j-1}, \mu) = 0,
$$

which, under Assumption 3.3, yields

$$
\lim_{j \in \mathcal{J}_1} \pi(x_{k_j}, y_{k_j-1}, \mu) = y_*. \tag{3.88}
$$

It then follows from this fact and (3.85) that

$$
\lim_{j \in \mathcal{J}_1} y_{k_j-1} = y_*. \tag{3.89}
$$

Combining (3.88) and (3.89) with (3.83) implies that

$$
\lim_{j \in \mathcal{J}_1} \widehat{y}_{k_j} = y_* \tag{3.90}
$$

so that $(x_*, y_*) = \lim_{j \in \mathcal{J}_1} (x_{k_j}, \widehat{y}_{k_j})$ is a first-order KKT point for problem (2.1). Finally, combining (3.90), the fact that the upper bound on the multipliers increases to infinity in (3.86), and the definition of y_{k+1} in line 37 of Algorithm 4, we have

$$
\lim_{j \in \mathcal{J}_1} y_{k_j} = y_*.
$$

Defining $\mathcal{J} := \mathcal{J}_1$, this completes the proof for this case. □

Finally, we consider the case when the penalty parameter converges to zero. For this case, we require the following technical lemma.

Lemma 3.12 *Suppose $l \leq x \leq u$ and let v be any vector in $\mathbb{R}^n$. Then, for any scalar $\xi > 1$ we have*

$$
\| P[x + \xi v] - x \|_2 \leq \xi \| P[x + v] - x \|_2.
$$

Proof It suffices to prove the result for the case when $l \leq 0 \leq u$ and $x = 0$ since the proof for the more general case is similar. We may write

$$
\begin{aligned}
P[\xi v] &= P[v] + (P[\xi v] - P[v]) =: P[v] + w_1 \\
\text{and } \xi P[v] &= P[v] + (\xi - 1) P[v] =: P[v] + w_2,
\end{aligned}
$$

where, since $\xi > 1$, we have for all $i \in \{1, \ldots, n\}$ that

$$[w_1]_i = \begin{cases} 0 & \text{if } \ v_i \le l_i \\ 0 & \text{if } v_i \ge u_i \\ l_i - v_i & \text{if } \ l_i < v_i < u_i \ \text{ and } \ \xi v_i \le l_i \\ u_i - v_i & \text{if } \ l_i < v_i < u_i \ \text{ and } \ \xi v_i \ge u_i \\ (\xi - 1)v_i & \text{if } \ l_i < \xi v_i < u_i \end{cases} \quad \text{and}$$

$$[w_2]_i = \begin{cases} (\xi - 1)l_i & \text{if } \ v_i \le l_i \\ (\xi - 1)u_i & \text{if } \ v_i \ge u_i \\ (\xi - 1)v_i & \text{if } \ l_i < v_i < u_i \ \text{ and } \ \xi v_i \le l_i \\ (\xi - 1)v_i & \text{if } \ l_i < v_i < u_i \ \text{ and } \ \xi v_i \ge u_i \\ (\xi - 1)v_i & \text{if } \ l_i < \xi v_i < u_i . \end{cases}$$

Hence, it is easily verified that

$$P[v]^T w_1 \le P[v]^T w_2 \quad \text{and} \quad \|w_1\|_2^2 \le \|w_2\|_2^2,$$

which along with the identity $\frac{1}{2}\|P[v]+w\|_2^2 = \frac{1}{2}\|P[v]\|_2^2 + P[v]^T w + \frac{1}{2}\|w\|_2^2$ yields the desired result. □

We now prove the following lemma, which reveals that if there are an infinite number of multiplier updates and the penalty parameter converges to zero, then the constraint violation measure converges to zero. Moreover, in such cases, as long as the number of decreases of the penalty parameter between consecutive multiplier updates is bounded, then any limit point of one of two possible subsequences is a first-order optimal point for (2.1).

Lemma 3.13 *If $|\mathcal{Y}| = \infty$ and $\mu_k \to 0$, then*

$$\lim_{k\to\infty} c_k = 0. \tag{3.91}$$

If, in addition, there exists a positive integer p such that $\mu_{k_j-1} \ge \gamma_\mu^p \mu_{k_{j-1}-1}$ for all sufficiently large j, then there exists an infinite ordered set $\mathcal{J} \subseteq \mathbb{N}$ such that

$$\lim_{j\in\mathcal{J}, j\to\infty} \|F_L(x_{k_j}, \widehat{y}_{k_j})\|_2 = 0 \quad \text{or} \quad \lim_{j\in\mathcal{J}, j\to\infty} \|F_L(x_{k_j}, \pi(x_{k_j}, y_{k_j-1}, \mu_{k_j-1}))\|_2 = 0. \tag{3.92}$$

In such cases, if the first (respectively, second) limit in (3.92) holds, then along with (3.91) it follows that any limit point of $\{(x_{k_j}, \widehat{y}_{k_j})\}_{j\in\mathcal{J}}$ (respectively, $\{(x_{k_j}, y_{k_j-1})\}_{j\in\mathcal{J}}$) is a first-order optimal point for problem (2.1).

Proof It follows from (3.40) that

$$\lim_{k\to\infty} \|c_k\|_2 = \bar{c} \ge 0. \tag{3.93}$$

However, it also follows from (3.76) and line 32 of Algorithm 4 that

$$\lim_{j\to\infty} \|c_{k_j}\|_2 \leq \lim_{j\to\infty} t_j = 0. \tag{3.94}$$

The limit (3.91) now follows from (3.93) and (3.94).

To prove the remainder of the result, first note that by lines 34 and 36 of Algorithm 4 and since

$$0 = \lim_{k\to\infty} \mu_k = \lim_{j\to\infty} \mu_{k_j}, \tag{3.95}$$

we have for all sufficiently large j that

$$\min\{\|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2, \|F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})\|_2\} \leq T_j = \gamma_T \mu_{k_{j-1}-1} T_{j-1}. \tag{3.96}$$

If there exists an infinite ordered set $\mathcal{J} \subseteq \mathbb{N}$ such that $\lim_{j\in\mathcal{J}, j\to\infty} \|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2 = 0$, then the first limit in (3.92) holds and there is nothing left to prove. Thus, suppose that $\{\|F_{\text{L}}(x_{k_j}, \widehat{y}_{k_j})\|_2\}$ is bounded below and away from zero for all sufficiently large j. Then, from (3.95) and (3.96), we have

$$\|F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})\|_2 \leq \gamma_T \mu_{k_{j-1}-1} T_{j-1}$$

for all sufficiently large j, from which it follows along with Lemma 3.12 that

$$\begin{aligned}
\gamma_T T_{j-1} &\geq \tfrac{1}{\mu_{k_{j-1}-1}} \|F_{\text{AL}}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})\|_2 \\
&\geq \|P[x_{k_j} - \tfrac{1}{\mu_{k_{j-1}-1}} \nabla_x \mathcal{L}(x_{k_j}, y_{k_j-1}, \mu_{k_j-1})] - x_{k_j}\|_2 \\
&= \|P[x_{k_j} - \tfrac{\mu_{k_j-1}}{\mu_{k_{j-1}-1}} (g(x_{k_j}) - J(x_{k_j})^T \pi(x_{k_j}, y_{k_j-1}, \mu_{k_j-1}))] - x_{k_j}\|_2.
\end{aligned}$$

With these inequalities, (3.76), and the fact that $\mu_{k_j-1}/\mu_{k_{j-1}-1} \in [\gamma_\mu^p, 1]$ for all sufficiently large j, Lemma 3.1 (taking a further infinite subset of $\mathcal{J}$, if necessary) yields the second limit in (3.92). □

3.2.3 Overall global convergence result

We combine the lemmas in the previous subsections to obtain the following result.

Theorem 3.14 *One of the following must hold true:*

(i) every limit point of $\{x_k\}$ is an infeasible stationary point;
(ii) $\mu_k \nrightarrow 0$ and there exists an infinite ordered set $\mathcal{K} \subseteq \mathbb{N}$ such that every limit point of $\{(x_k, \widehat{y}_k)\}_{k\in\mathcal{K}}$ is first-order optimal for (2.1); or
(iii) $\mu_k \to 0$, every limit point of $\{x_k\}$ is feasible, and if there exists a positive integer p such that $\mu_{k_j-1} \geq \gamma_\mu^p \mu_{k_{j-1}-1}$ for all sufficiently large j, then there exists an infinite ordered set $\mathcal{J} \subseteq \mathbb{N}$ such that any limit point of either $\{(x_{k_j}, \widehat{y}_{k_j})\}_{j\in\mathcal{J}}$ or $\{(x_{k_j}, y_{k_j-1})\}_{j\in\mathcal{J}}$ is first-order optimal for problem (2.1).

Proof Lemmas 3.8, 3.9, 3.10, and 3.13 cover the only four possible outcomes of Algorithm 4; the result follows from those described in these lemmas. □

We complete this section with a discussion of Theorem 3.14. As with all penalty methods for nonconvex optimization, Algorithm 4 may converge to a local minimizer of the constraint violation that is not feasible, i.e., an infeasible stationary point. This possible outcome is, in fact, unavoidable since we have not (implicitly) assumed that problem (2.1) is feasible. By far, the most common outcome of our numerical results in Sect. 4 is case (ii) of Theorem 3.14, in which the penalty parameter ultimately remains fixed and convergence to a first-order primal-dual solution of (2.1) is observed. In fact, these numerical tests show that our adaptive algorithm is far more efficient than our basic implementation and, importantly, at least as reliable. The final possible outcome is that the penalty parameter and the constraint violation both converge to zero. In this case we are unable to guarantee that limit points are first-order solutions of (2.1), even under the assumption that the MFCQ holds, and therefore have obtained a weaker convergence result than many other augmented Lagrangian methods. Nonetheless, we remain content since the numerical tests in Sect. 4 show that Algorithm 4 is superior to the basic approach and that the penalty parameter consistently remains bounded away from zero.

Of course, it is possible to view our adaptive strategy as a mechanism for quickly obtaining an improved Lagrange multiplier estimate and value for the penalty parameter. These values may then be used as the initial input for a traditional augmented Lagrangian method. For example, motivated by Lemma 3.13, one could employ our algorithm, but transition to a traditional augmented Lagrangian method once the Lagrange multiplier estimate has been updated more than a prescribed number of times, the quantities μ_k and $\|c(x_k)\|$ are below prescribed positive tolerances, and the penalty parameter has been decreased more than a prescribed number of times since the most recent Lagrange multiplier estimate update. This simple strategy inherits the well-documented convergence theory for standard augmented Lagrangian methods and benefits from the practical advantages of steering exhibited by our approach.

Finally, it is also possible to modify Algorithm 4 so that convergence to first-order optimal points may be established even in case (iii) of Theorem 3.14. Specifically, we could make the following changes: (i) compute first-order multiplier estimates $\widehat{y}_{k+1}$ during *every* iteration (whereas currently they are only computed when $\|c_{k+1}\|_2 \leq t_j$); (ii) switch the order of the two **if** statements in Lines 32 and 34 of Algorithm 4; (iii) explicitly limit the number of decreases of the penalty parameter allowed between updates to the multiplier vector (as motivated by part (iii) of Theorem 3.14); and (iv) if the explicit bound on the number of updates allowed by part (iii) is reached, then do not allow a further decrease to the penalty parameter until either the multiplier is updated again, or an approximate minimizer of the augmented Lagrangian is found at which the constraint violation is not sufficiently small, i.e., not less than t_j. Although these changes could be made to Algorithm 4, we have chosen not to do so for two reasons. First, these additions would further complicate the algorithm in a manner that we do not believe is justified from a practical perspective. Second, again from a practical perspective, it may be inefficient to compute new multiplier estimates in every iteration.

4 Numerical experiments

In this section we describe the details of an implementation of our trust region method and provide the results of numerical experiments. These experiments include three parts. First, we provide a single illustrative example to highlight the benefits of our adaptive approach. Second, we compare our method with Lancelot [10] on a subset of the equality constrained CUTEr [26] test problems. Third, we contrast our basic and adaptive augmented Langrangian methods on a subset of inequality constrained Hock-Schittkowski test problems [28]. All together, we believe that these experiments are the best way to isolate and measure the performance of our adaptive penalty parameter updating procedure.

4.1 Implementation details

Algorithm 4 was implemented in MATLAB and, hereinafter, will be referred to as `AAL-TR`. For comparison purposes, we also implemented a trust-region variant of the basic augmented Lagrangian method given by Algorithm 1, which we will refer to as `BAL-TR`. The components of these two implementations were mostly identical. For example, the implemented update for the trust region radii and trial step acceptance procedure were exactly the same.

A critical component of both implementations was the approximate solution of the trust region subproblem (3.8) and, in the case of `AAL-TR`, subproblem (3.4). For the equality constrained problems of Sect. 4.3, we approximately solved these subproblems (defined by an ℓ_2 trust-region constraint) with an implementation of a conjugate gradient (CG) algorithm by Steihaug/Toint [11,13]. (In this manner, the Cauchy points for each subproblem were obtained in the first iteration of CG.) For each subproblem for which it was employed, the CG iteration was run until either the trust region boundary was reached (perhaps due to a negative curvature direction being obtained) or the gradient of the subproblem objective was below a predefined constant $\kappa_{cg} > 0$. In the case of `BAL-TR`, the solution obtained from CG was used in a minor iteration—see step 11 of Algorithm 1—whereas in the case of `AAL-TR`, the solutions obtained from CG were used as explicitly stated in Algorithm 4. We believe this strategy allows for a fair comparison with `Lancelot` on equality constrained problems since `Lancelot` employs a similar iterative scheme for the subproblem solves. However, for the inequality constrained problems tested in Sect. 4.4, we only perform comparisons between `BAL-TR` and `AAL-TR`. In this setting, we simply use the CPLEX QP solver [29] to compute "exact" solutions to the subproblems (defined by an ℓ_∞ trust-region constraint), but with an infinity norm trust-region so that they are bound-constrained QPs. Since CPLEX requires the QP to be convex, we add multiples of ten times the identity matrix to the Hessian of the QPs until CPLEX successfully returns a solution without encountering negative curvature. Importantly, this simple strategy was used for both our basic and adaptive strategies to allow for a fair comparison. (We do not compare with `Lancelot` on inequality constrained problems as it employs an iterative scheme to solve possibly indefinite QP subproblems. Such a striking difference in the manner in which the subproblems are solved would lead to

Table 1 Input parameter values used in AAL-TR and BAL-TR

Par.	Val.	Par.	Val.	Par.	Val.	Par.	Val.
γ_μ	5e−01	κ_3	1e−04	Y	∞	Y_1	∞
γ_t	5e−01	κ_t	9e−01	Γ_δ	6e−01	κ_{cg}	1e−10
γ_T	5e−01	η_s	1e−02	μ_0	1e+00	κ_{opt}	1e−06
γ_δ	5e−01	η_{vs}	9e−01	t_0	1e+00	κ_{feas}	1e−06
κ_F	9e−01	δ	1e+04	t_1	1e+00	$\mu_{\min}$	1e−10
κ_1	1e+00	δ_R	1e−04	δ_0	1e+00	$k_{\max}$	1e+03
κ_2	1e+00	ϵ	5e−01	T_1	1e+00	G	1e+02

a comparison between AAL-TR and Lancelot that does not isolate the effect that steering has on efficiently finding solutions.)

A second component of AAL-TR that requires specification is the computation of the estimates $\{\widehat{y}_{k+1}\}$ to satisfy (3.13). If $\|F_{\text{L}}(x_{k+1}, \pi(x_{k+1}, y_k, \mu_k))\|_2 \leq \|F_{\text{L}}(x_{k+1}, y_k)\|_2$, then we set $\widehat{y}_{k+1} \leftarrow \pi(x_{k+1}, y_k, \mu_k)$; otherwise, we set $\widehat{y}_{k+1} \leftarrow y_k$.

Algorithms AAL-TR and BAL-TR both terminated with a declaration of optimality if

$$\|F_{\text{L}}(x_k, y_k)\|_\infty \leq \kappa_{\text{opt}} \quad \text{and} \quad \|c_k\|_\infty \leq \kappa_{\text{feas}}, \tag{4.1}$$

and terminated with a declaration that an infeasible stationary point was found if

$$\|F_{\text{FEAS}}(x_k)\|_\infty \leq \kappa_{\text{opt}}/10, \quad \|c_k\|_\infty > \kappa_{\text{feas}}, \quad \text{and} \quad \mu_k \leq \mu_{\min}. \tag{4.2}$$

Note that in the latter case our implementations differ slightly from Algorithms 1 and 4 as we did not declare that an infeasible stationary point was found until the penalty parameter was below a prescribed tolerance. The motivation for this was to avoid premature termination at (perhaps only slightly) infeasible points at which the gradient of the infeasibility measure $\|F_{\text{FEAS}}(x_k)\|_\infty$ was relatively small compared to $\|c_k\|_\infty$. Also, each algorithm terminated with a declaration of failure if neither (4.1) nor (4.2) was satisfied within an iteration limit $k_{\max}$. The problem functions were prescaled so that the ℓ_∞-norms of the gradients of each at the initial point would be less than or equal to a prescribed constant $G > 0$. This helped to facilitate termination for poorly-scaled problems.

Table 1 summarizes the input parameter values that were chosen. Note that our choice for Y means that we did not impose explicit bounds on the norms of the multipliers, meaning that we effectively always chose $\alpha_y \leftarrow 1$ in (3.14).

4.2 An illustrative example

To illustrate the benefits of our adaptive updating strategy for the penalty parameter, we study the CUTEr problem CATENA. We have chosen a relatively small instance of the problem that consists of 15 variables and 4 equality constraints. We compare our results with that obtained by Lancelot, a very mature Fortran-90 implementation

Table 2 Input parameter values used in `Lancelot`

Parameter	Value
HISTORY-LENGTH-FOR-NON-MONOTONE-DESCENT	0.0
MAGICAL-STEPS-ALLOWED	NO
USE-TWO-NORM-TRUST-REGION	YES
SUBPROBLEM-SOLVED-ACCURATELY	NO
INITIAL-PENALTY-PARAMETER	1.0

whose overall approach is well-represented by Algorithm 1. In fact, the algorithm in `Lancelot` benefits from a variety of advanced features from which the implementation of our methods could also benefit. However, since ours are only preliminary implementations, we set input parameters for `Lancelot` as described in Table 2

```
Iter  #g.ev       f     proj.g  penalty   target   infeas  y
   0      1  -5.89E+03  1.2E+03  1.0E+00
   1      1  -5.89E+03  1.2E+03
   2      2  -1.38E+04  1.2E+03
   3      3  -2.88E+04  1.2E+03
   4      4  -5.07E+04  1.6E+03
   5      5  -6.79E+04  1.6E+03
   6      6  -7.92E+04  1.8E+03
   7      7  -7.94E+04  2.8E+03
   8      8  -8.43E+04  5.8E+02
   9      9  -8.48E+04  9.1E+01
  10     10  -8.49E+04  4.0E+00
  11     11  -8.49E+04  8.6E-03           1.0E-01  1.7E+02
  12     12  -1.21E+04  3.0E+03  1.0E-01
  13     13  -3.79E+04  6.5E+02
  14     14  -3.99E+04  7.1E+01
  15     15  -3.99E+04  1.3E+00
  16     16  -3.99E+04  4.5E-04           1.0E-01  3.6E+01
  17     17  -7.08E+03  3.0E+03  1.0E-02
  18     18  -1.88E+04  6.4E+02
  19     19  -1.96E+04  6.9E+01
  20     20  -1.96E+04  1.2E+00
  21     21  -1.96E+04  3.7E-04           7.9E-02  7.4E+00
  22     22  -6.19E+03  2.9E+03  1.0E-03
  23     23  -1.10E+04  6.0E+02
  24     24  -1.13E+04  5.7E+01
  25     25  -1.13E+04  7.1E-01
  26     26  -1.13E+04  1.1E-04           6.3E-02  1.4E+00
  27     27  -7.57E+03  2.6E+03  1.0E-04
  28     28  -8.79E+03  3.9E+02
  29     29  -8.82E+03  1.4E+01
  30     30  -8.82E+03  1.7E-02
  31     31  -8.82E+03  7.0E-08           5.0E-02  2.1E-01
  32     32  -8.36E+03  1.4E+03  1.0E-05
  33     33  -8.40E+03  2.8E+01
  34     34  -8.40E+03  1.2E-02
  35     35  -8.40E+03  1.0E-06           4.0E-02  2.3E-02  1
  36     36  -8.30E+03  2.7E+01
  37     37  -8.30E+03  1.2E-02
  38     38  -8.30E+03  2.8E-09           1.3E-06  2.9E-04
  39     39  -8.34E+03  5.0E-02  1.0E-06
  40     40  -8.34E+03  3.7E-06
  41     41  -8.34E+03  2.1E-10           3.2E-02  3.0E-05  1
  42     42  -8.34E+03  6.4E-04
  43     43  -8.34E+03  1.4E-09
```

Fig. 1 Output from `Lancelot` for problem `CATENA`

to have as fair a comparison as possible. The first two parameters disallowed a non-monotone strategy and the possibility of using so-called "magical steps". The third and fourth parameters ensured that an ℓ_2-norm trust region constraint was used and allowed the possibility of inexact subproblem solves, roughly as described in Sect. 4.1. The last input parameter ensured that the initial value of the penalty parameter was the same as for our implementation (see Table 1).

With the above alterations to its default parameters, we solved `CATENA` using `Lancelot`. A stripped version of the output is given in Fig. 1. The columns represent the iteration number (`Iter`), cumulative number of gradient evaluations (`#g.ev`), objective value (`f`), projected gradient norm (`proj.g`), penalty parameter value (`penalty`), constraint violation target (`target`), constraint violation value (`infeas`), and a flag for which a value of 1 indicates that a multiplier update occurred. (An empty entry indicates that a value did not change since the previous iteration.) We make a couple observations. First, `Lancelot` only attempted to update the penalty parameter and multiplier vector on iterations 11, 16, 21, 26, 31, 35, 38, 41, and 43 when an approximate minimizer of the augmented Lagrangian was identified. Second, the constraint violation target was only satisfied in iterations 35, 41, and 43. Thus, one might view the iterations that only lead to a decrease in the penalty parameter as ineffective since the target in feasibility was not obtained and the only (possible) progress may have been in the primal space. For this example, this includes iterations 1–31 and 36–38, which accounted for $\approx$79% of the iterations.

Next we solved `CATENA` using `AAL-TR` and provide the results in Fig. 2. The columns represent the iteration number (`Iter.`), objective value (`Objective`), a measure of infeasibility (`Infeas.`), $\|g_k - J_k^T y_k\|_\infty$ (`Lag.Err.`), the target constraint violation for the current value of the penalty parameter and Lagrange multiplier vector (`Fea.Tar.`), the value of the penalty parameter (`Pen.Par.`), $\Delta q_v(r_k; x_k)$ (`Dqv(r)`), $\Delta q_v(s_k; x_k)$ (`Dqv(s)`), and a flag for which a value of 1 indicates that a multiplier update occurred (`y`).

One can see that our method solved the problem efficiently since it quickly realized that a decrease in the penalty parameter would be beneficial. Moreover, the feasibility measure `Fea.Err.` and optimality measure `Lag.Err.` converged quickly. This was due, in part, to the fact that the multiplier vector was updated during most iterations near the end of the run. In particular, `AAL-TR` updated the multiplier vector during 5 of the last 6 iterations.

The most instructive column for witnessing the benefits of our penalty parameter updating strategy is `Dqv(s)` since this column shows the predicted decrease in the constraint violation yielded by the trial step s_k. When this quantity was positive the trial step predicted progress toward constraint satisfaction, and when it was negative the trial step predicted an increase in constraint violation. This quantity was compared with the quantity in column `Dqv(r)` in the steering condition (3.11c). It is clear in the output that the penalty parameter was decreased when the steering step predicted an increase in constraint violation, i.e., when `Dqv(s)` was negative, though exceptions were made when the constraint violation was well within `Fea.Tar.`, the constraint violation target.

This particular problem shows that our adaptive strategy has the potential to be very effective on problems that require the penalty parameter to be reduced from its initial

Iter.	Objective	Infeas.	Lag.Err.	Fea.Tar.	Pen.Par.	Dqv(r)	Dqv(s)	y
0	-5.89e+03	2.80e-01	1.00e+00	2.52e-01	1.00e+00	+1.52e-01	-1.82e-01	
					5.00e-01		-1.81e-01	
					2.50e-01		-1.77e-01	
					1.25e-01		-1.71e-01	
					6.25e-02		-1.59e-01	
					3.12e-02		-1.35e-01	
					1.56e-02		-9.35e-02	
					7.81e-03		-2.82e-02	
					3.91e-03		+3.98e-02	
1	-7.94e+03	4.71e-01	3.91e-03	2.52e-01	3.91e-03	+1.91e-01	+7.72e-02	
2	-9.42e+03	6.31e-01	3.91e-03	2.52e-01	3.91e-03	+2.44e-01	-1.11e-02	
					1.95e-03		+2.43e-01	
3	-9.42e+03	6.31e-01	1.95e-03	2.52e-01	1.95e-03	+2.37e-01	+2.06e-01	
4	-9.53e+03	3.01e-01	1.95e-03	2.52e-01	1.95e-03	+8.30e-02	+1.48e-02	
5	-9.53e+03	3.01e-01	1.95e-03	2.52e-01	1.95e-03	+5.09e-02	+1.15e-02	
6	-9.68e+03	3.22e-01	1.95e-03	2.52e-01	1.95e-03	+6.95e-02	+5.00e-02	
7	-9.68e+03	3.22e-01	1.95e-03	2.52e-01	1.95e-03	+4.00e-02	+3.22e-02	
8	-9.62e+03	2.83e-01	1.95e-03	2.52e-01	1.95e-03	+2.70e-02	-1.44e-02	
					9.77e-04		+2.98e-02	1
9	-9.40e+03	2.25e-01	9.77e-04	1.26e-01	9.77e-04	+3.62e-02	+3.78e-02	
10	-9.05e+03	1.64e-01	9.77e-04	1.26e-01	9.77e-04	+1.80e-02	-2.77e-03	
					4.88e-04		+2.02e-02	1
11	-8.74e+03	9.13e-02	4.88e-04	4.47e-02	4.88e-04	+1.06e-02	+8.95e-03	
12	-8.74e+03	9.13e-02	4.88e-04	4.47e-02	4.88e-04	+7.09e-03	+4.46e-03	
13	-8.71e+03	9.13e-02	4.88e-04	4.47e-02	4.88e-04	+6.05e-03	-1.11e-03	
					2.44e-04		+6.17e-03	1
14	-8.55e+03	4.42e-02	3.94e-05	9.46e-03	2.44e-04	+2.03e-03	+2.02e-03	1
15	-8.36e+03	1.75e-03	1.09e-05	9.20e-04	2.44e-04	+2.71e-06	+1.76e-06	
16	-8.35e+03	9.90e-04	1.07e-05	9.20e-04	2.44e-04	+9.36e-07	-9.78e-09	
					1.22e-04		+6.94e-07	1
17	-8.35e+03	5.01e-04	1.90e-09	2.79e-05	1.22e-04	+2.42e-07	+2.42e-07	1
18	-8.35e+03	4.74e-06	1.01e-09	1.47e-07	1.22e-04	+2.80e-11	+2.80e-11	1
19	-8.35e+03	1.42e-07	8.84e-14	1.00e-07	1.22e-04	---------	---------	-

Fig. 2 Output from `AAL-TR` for problem `CATENA`

value. In the next section we test the effectiveness of our new adaptive strategy on a large subset of equality constrained problems from the CUTEr test set.

4.3 Equality constrained CUTEr test problems

In this section we observe the effects of our penalty parameter updating strategy on a subset of the equality constrained CUTEr [26] test problems. We obtained our subset by first choosing all constrained problems with equality constraints only. Next, we eliminated *aug2dc* and *dtoc3* because they are quadratic problems that were too large for our MATLAB implementation to solve. Next, we eliminated *argtrig, artif, bdvalue, bdvalues, booth, bratu2d, bratu2dt, brownale, broydn3d, cbratu2d, cbratu3d, chandheu, chnrsbne, cluster, coolhans, cubene, drcavty1, drcavty2, drcavty3, eigenau, eigenb, eigenc, flosp2th, flosp2tl, flosp2tm, gottfr, hatfldf, hatfldg, heart8, himmelba, himmelbc, himmelbe, hypcir, integreq, msqrta, msqrtb, powellbs, powellsq, recipe, rsnbrne, sinvalne, spmsqrt, trigger, yatp1sq, yatp2sq, yfitne*, and *zangwil3* because `Lancelot` recognized them as not having an objective function, in which case a penalty parameter was not required. This is appropriate since otherwise we would

not be comparing a traditional AL method to our adaptive AL method, which is the purpose of these tests. Next, we removed *heart6, hydcar20, hydcar6, methanb8,* and *methanl8* because these are nonlinear equation solving problems and `Lancelot` solved them without introducing a penalty parameter. Again, since we wish to compare our adaptive AL method to a traditional AL method, we feel this is justified. Finally, we removed *arglcle, junkturn,* and *woodsne* because all of the solvers (including `Lancelot`) converged to a point that was recognized as an infeasible stationary point: on problem *arglcle* `Lancelot` terminated in 6 iterations and `AAL-TR` terminated in 3 iterations; on problem *junkturn* `Lancelot` terminated in 117 iterations and `AAL-TR` terminated in 312 iterations; and on problem *woodsne* `Lancelot` terminated in 103 iterations and `AAL-TR` terminated in 67 iterations. This left us with a total of 91 problems composing our subset: *bt1, bt10, bt11, bt12, bt2, bt3, bt4, bt5, bt6, bt7, bt8, bt9, byrdsphr, catena, chain, dixchlng, dtoc1l, dtoc2, dtoc4, dtoc5, dtoc6, eigena2, eigenaco, eigenb2, eigenbco, eigenc2, eigencco, elec, genhs28, gridnetb, gridnete, gridneth, hager1, hager2, hager3, hs100lnp, hs111lnp, hs26, hs27, hs28, hs39, hs40, hs42, hs46, hs47, hs48, hs49, hs50, hs51, hs52, hs56, hs6, hs61, hs7, hs77, hs78, hs79, hs8, hs9, lch, lukvle1, lukvle10, lukvle11, lukvle12, lukvle13, lukvle14, lukvle15, lukvle16, lukvle17, lukvle18, lukvle2, lukvle3, lukvle4, lukvle6, lukvle7, lukvle8, lukvle9, maratos, mss1, mwright, optctrl3, optctrl6, orthrdm2, orthrds2, orthrega, orthregb, orthregc, orthregd, orthrgdm, orthrgds,* and *s316-322*.

We compare the performance of `AAL-TR`, `BAL-TR`, and `Lancelot` on this set of problems. As previously mentioned, both `Lancelot` and `BAL-TR` are based on Algorithm 1, though the details of their implementations are quite different. Therefore, we can compare them to illustrate that our implementation of `BAL-TR` is roughly on par with `Lancelot` for solving equality constrained problems in terms of performance.

To measure performance, we use performance profiles as introduced by Dolan and Moré [16]. They provide a concise mechanism for comparing algorithms over a collection of problems. Roughly, a performance profile chooses a relative metric, e.g., the number of iterations, and then plots the fraction of problems (y-axis) that are solved within a given factor (x-axis) of the best algorithm according to this metric. Roughly speaking, more robust algorithms are "on top" towards the right side of the plot, and more efficient algorithms are "on top" near the left side of the plot. Thus, algorithms that are "on top" throughout the plot may be considered more efficient and more robust, which is the preferred outcome.

The performance profiles in Figs. 3 and 4 compare `AAL-TR`, `BAL-TR`, and `Lancelot` in terms of iterations and gradient evaluations, respectively. (Note that in this comparison, we compare the number of "minor" iterations in `Lancelot` with the number of iterations in our methods.) These figures show that `BAL-TR` performs similarly to `Lancelot` in terms of iterations (Fig. 3) and gradient evaluations (Fig. 4). Moreover, it is also clear that our adaptive penalty parameter updating scheme in `AAL-TR` yields better efficiency and reliability when compared to `BAL-TR` and `Lancelot` on this collection of problems.

Finally, it is pertinent to compare the final value of the penalty parameter for `Lancelot` and our adaptive algorithm. We present these outcomes in Table 3. The

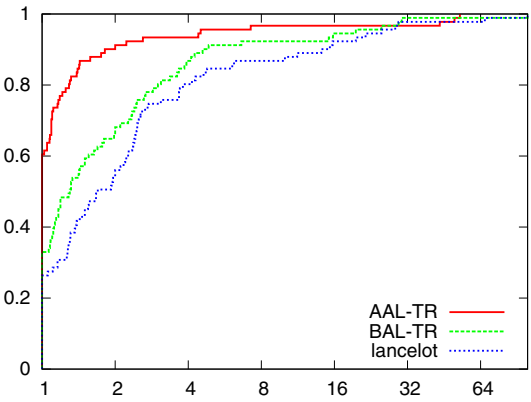

Fig. 3 Performance profile for iterations comparing `AAL-TR`, `BAL-TR`, and `Lancelot`

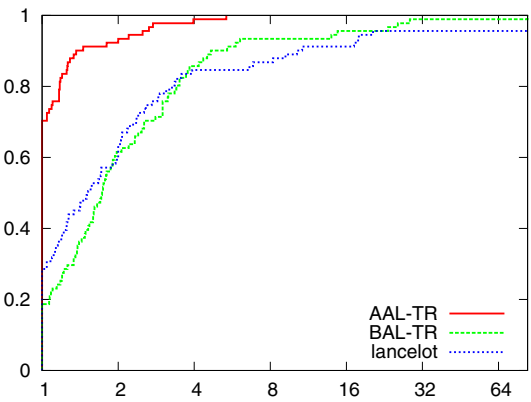

Fig. 4 Performance profile for gradient evaluations comparing `AAL-TR`, `BAL-TR`, and `Lancelot`

Table 3 Numbers of CUTEr problems for which the final penalty parameter value was in the given ranges

μ_{final}	`Lancelot`	Algorithm 4	μ_{final}	`Lancelot`	Algorithm 4
1	7	15	$[10^{-4}, 10^{-3})$	7	10
$[10^{-1}, 1)$	34	7	$[10^{-5}, 10^{-4})$	5	6
$[10^{-2}, 10^{-1})$	16	18	$[10^{-6}, 10^{-5})$	2	4
$[10^{-3}, 10^{-2})$	13	20	$(0, 10^{-6})$	7	11

column μ_{final} gives ranges for the final value of the penalty parameter, while the remaining columns give the numbers of problems out of the total 91 whose final penalty parameter fell within the specified range.

We make two observations about the results provided in Table 3. First, we observe that our adaptive updating strategy generally does not drive the penalty parameter smaller than does `Lancelot`. This is encouraging since the traditional updating approach used in `Lancelot` is very conservative, so it appears that our adaptive strategy obtains superior performance without driving the penalty parameter to unnec-

essarily small values. Second, we observe that our strategy maintains the penalty parameter at its initial value ($\mu_0 \leftarrow 1$) on more problems than does `Lancelot`. This phenomenon can be explained by considering the situations in which `Lancelot` and our methods update the penalty parameter. `Lancelot` considers an update once the augmented Lagrangian has been minimized for $y = y_k$ and $\mu = \mu_k$. If the constraint violation is too large and/or the Lagrange multipliers are poor estimates of the optimal multipliers—both of which may easily occur at the initial point—then the penalty parameter is decreased if/when such a minimizer of the augmented Lagrangian is not sufficiently close to the feasible region. That is, `Lancelot` looks *globally* at the progress towards feasibility obtained from a given point to the next minimizer of the augmented Lagrangian. Our method, on the other hand, observes the *local* reduction in linearized constraint violation. As long as this local reduction is sufficiently large, then no reduction of the penalty parameter occurs, no matter the progress towards nonlinear constraint satisfaction that has occurred so far. Overall, we feel that this behavior illustrated in Table 3 highlights a strength of our algorithm, which is that it is less sensitive to the nonlinear constraint violation targets than is `Lancelot`.

4.4 Inequality constrained Hock-Schittkowski test problems

In this section we test our penalty parameter updating strategy by comparing our basic and adaptive algorithms `BAL-TR` and `AAL-TR`, respectively, on a subset of the inequality constrained Hock-Schittkowski test problems [28]. This subset was obtained by first choosing all problems with at least one general inequality constraint. Next, we eliminated *hs21*, *hs35*, *hs44*, *hs53*, *hs76*, and *hs118* because they are simple quadratic problems, i.e., have quadratic objective functions and linear constraints. We also eliminated *hs67*, *hs85*, and *hs87* because they are not smooth optimization problems. Next, we eliminated problem *hs13* because the linear independent constraint qualification is not satisfied at the solution. Problems *hs72*, *hs88*, *hs89*, *hs90*, *hs91*, and *hs92* were also removed since both `BAL-TR` and `AAL-TR` converged to an infeasible stationary point. Finally, we removed problems *hs101*, *hs102*, *hs103*, *hs109*, and *hs116* since both `BAL-TR` and `AAL-TR` did not reach the desired optimality tolerances. This left us with the following collection of 64 test problems: *hs10, hs11, hs12, hs14, hs15, hs16, hs17, hs18, hs19, hs20, hs22, hs23, hs24, hs29, hs30, hs31, hs32, hs33, hs34, hs36, hs37, hs41, hs43, hs54, hs55, hs57, hs59, hs60, hs62, hs63, hs64, hs65, hs66, hs68, hs69, hs70, hs71, hs73, hs74, hs75, hs76, hs80, hs81, hs83, hs84, hs86, hs93, hs95, hs96, hs97, hs98, hs99, hs100, hs104, hs105, hs106, hs107, hs108, hs111, hs112, hs113, hs114, hs117, and hs119.*

The results on the previous test set are given in the form of performance profiles in Figs. 5 and 6, and Table 4 (see Sect. 4.3 for a description of how these types of profiles should be understood). The two profiles clearly show that our adaptive trust-region algorithm is superior to a traditional approach in terms of efficiency without sacrificing reliability. These results are promising and are further complemented by Table 4, which shows that the final penalty parameter associated with our adaptive strategy does not drive the penalty parameter unnecessarily small.

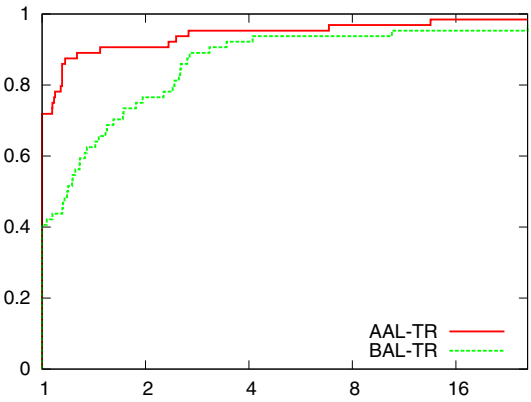

Fig. 5 Performance profile for iterations comparing AAL-TR and BAL-TR

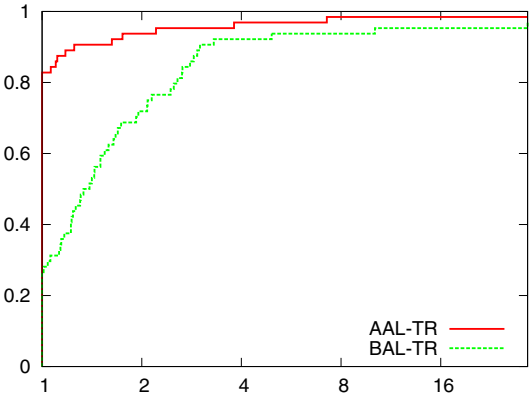

Fig. 6 Performance profile for gradient evaluations comparing AAL-TR and BAL-TR

Table 4 Numbers of Hock-Schittkowski problems for which the final penalty parameter value was in the given ranges

μ_{final}	AAL-TR	BAL-TR	μ_{final}	AAL-TR	BAL-TR
1	11	17	$[10^{-4}, 10^{-3})$	7	7
$[10^{-1}, 1)$	5	12	$[10^{-5}, 10^{-4})$	8	2
$[10^{-2}, 10^{-1})$	13	9	$[10^{-6}, 10^{-5})$	2	1
$[10^{-3}, 10^{-2})$	17	13	$(0, 10^{-6})$	1	3

5 Conclusion

We have proposed, analyzed, and tested a new AL algorithm for large-scale constrained optimization. The novel feature of the algorithm is an adaptive strategy for updating the penalty parameter. We have proved that our algorithm is well-posed, possesses global convergence guarantees, and outperforms traditional AL methods in terms of the numbers of iterations and gradient evaluations on a wide range of test problems. These improvements are realized by requiring the computation of an additional steering

step as an approximate solution to a convex QP. The conditions that we require of the steering step allow for an approximate solution in the form of a simple Cauchy step whose extra computation is negligible. A potential disadvantage of this simple choice is that convergence may be slow on infeasible problems. In this case more accurate solutions to the feasibility subproblem are likely to accelerate convergence to locally infeasible minimizers of the constraint violation. A final observation is that the computation of the steering step and trial step may be performed in parallel, again making the additional calculations (in terms of time) negligible.

One potential deficiency of our proposed technique is the lack of a guarantee that the penalty parameter will remain bounded away from zero, even when it is applied to solve problems where a constraint qualification (e.g., MFCQ) is satisfied at all solution points. We stress that in our numerical experiments, we did not find that our method lead to unnecessary decreases in the penalty parameter, but this is an important matter to consider in general. Fortunately, one simple technique to avoid this issue is to transition from our algorithm to a traditional AL approach once consistent progress towards nonlinear constraint satisfaction is observed. In this fashion, the convergence guarantees of traditional AL methods can be relied upon in neighborhoods of solution points.

Acknowledgments The authors are grateful to the two referees and Associate Editor whose comments and suggestions helped to significantly improve the paper. They are also indebted to Nick Gould and Andrew Conn whose insights were extremely valuable in the preparation of this article.

References

1. Andreani, R., Birgin, E.G., Martínez, J.M., Schuverdt, M.L.: Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. Math. Program. **111**, 5–32 (2008)
2. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Computer Science and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York (1982)
3. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific, Belmont, Massachusetts (1996)
4. Birgin, E.G., Martínez, J.M.: Augmented Lagrangian method with nonmonotone penalty parameters for constrained optimization. Comput. Optim. Appl. **51**, 941–965 (2012)
5. Boggs, P.T., Tolle, J.W.: A family of descent functions for constrained optimization. SIAM J. Numer. Anal. **21**, 1146–1161 (1984)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn **3**, 1–122 (2011)
7. Byrd, R.H., Lopez-Calva, G., Nocedal, J.: A line search exact penalty method using steering rules. Math Program **133**, 39–73 (2012)
8. Byrd, R.H., Nocedal, J., Waltz, R.A.: Steering exact penalty methods for nonlinear programming. Optim. Methods Softw. **23**, 197–213 (2008)
9. Conn, A.R., Gould, N.I.M., Toint, P.L.: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. **28**, 545–572 (1991)
10. Conn, A.R., Gould, N.I.M., Toint, P.L.: LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A). Lecture Notes in Computation Mathematics, vol. 17. Springer, Berlin, Heidelberg, New York, London, Paris and Tokyo (1992)
11. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust-Region Methods. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2000)
12. Davidson, K.R., Donsig, A.P.: Real analysis and applications. Undergraduate Texts in Mathematics. Springer, New York (2010). Theory in practice

13. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. SIAM J. Numer. Anal. **19**, 400–408 (1982)
14. Dennis Jr, J., El-Alem, M., Maciel, M.C.: A global convergence theory for general trust-region-based algorithms for equality constrained optimization. SIAM J. Optim. **7**, 177–207 (1997)
15. DiPillo, G., Grippo, L.: A new class of augmented Lagrangians in nonlinear programming. SIAM J. Control Optim. **17**, 618–628 (1979)
16. Dolan, E.D., Moré, J.J.: Benchmarking Optimization Software with COPS, Technical Memorandum ANL/MCS-TM-246. Argonne National Laboratory, Argonne, IL (2000)
17. El-Alem, M.: A global convergence theory for dennis, el-alem, and maciel's class of trust-region algorithms for constrained optimization without assuming regularity. SIAM J. Optim. **9**, 965–990 (1999)
18. Fernández, D., Solodov, M.: Local convergence of exact and inexact augmented Lagrangian methods under the second-order sufficiency condition. IMPA, preprint A677 (2010)
19. Fernández, D., Solodov, M.: Stabilized sequential quadratic programming for optimization and a stabilized Newton-type method for variational problems. Math. Program. **125**, 47–73 (2010)
20. Fletcher, R.: A class of methods for nonlinear programming with termination and convergence properties. In: Abadie, J. (ed.) Integer and Nonlinear Programming, pp. 157–175. North-Holland, The Netherlands (1970)
21. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput. Math. Appl. **2**, 17–40 (1976)
22. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**, 99–131 (2005)
23. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: Some Theoretical Properties of an Augmented Lagrangian Merit Function, Report SOL 86–6R. Stanford University, Stanford, CA (1986)
24. Gill, P.E., Robinson, D.P.: A Globally Convergent Stabilized SQP Method, Numerical Analysis Report 12–02. University of California, San Diego, La Jolla, CA (2012)
25. Glowinski, R., Marroco, A.: Sur l'Approximation, par Elements Finis d'Ordre Un, el la Resolution, par Penalisation-Dualité, d'une Classe de Problèmes de Dirichlet Nonlineares. Revue Française d'Automatique, Informatique et Recherche Opérationelle **9**, 41–76 (1975)
26. Gould, N.I.M., Orban, D., Toint, P.L.: CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. **29**, 373–394 (2003)
27. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theory Appl. **4**, 303–320 (1969)
28. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, vol. 187. Springer, Berlin, Heidelberg and New York (1981)
29. ILOG Inc, ILOG CPLEX: High-performance software for mathematical programming and optimization, 2006. See http://www.ilog.com/products/cplex/
30. Izmailov, A.F., Solodov, M.V.: On attraction of linearly constrained Lagrangian methods and of stabilized and quasi-Newton SQP methods to critical multipliers. Math. Program. **126**, 231–257 (2011)
31. Izmailov, A.F., Solodov, M.V.: Stabilized SQP revisited. Math. Program. **133**, 93–120 (2012)
32. Kočvara, M., Stingl, M., PENNON: A generalized augmented Lagrangian method for semidefinite programming, in High performance algorithms and software for nonlinear optimization (Erice: vol. 82 of Appl. Optim., Kluwer Acad. Publ. Norwell, MA 2003, pp. 303–321 (2001)
33. Li, D.-H., Qi, L.: A stabilized SQP method via linear equations, technical Report AMR00/5. University of New South Wales, Sydney, School of Mathematics (2000)
34. Mongeau, M., Sartenaer, A.: Automatic decrease of the penalty parameter in exact penalty function methods. Eur. J. Oper. Res. **83**(3), 686–699 (1995)
35. Moré, J.J.: Trust regions and projected gradients. In M. Iri and K. Yajima, (eds.) System Modelling and Optimization. vol. 113 of Lecture Notes in Control and Information Sciences, pp. 1–13. Springer, Berlin Heidelberg (1988)
36. Mostafa, E.-S., Vicente, L., Wright, S.: Numerical behavior of a stabilized SQP method for degenerate NLP problems. In C. Bliek, C. Jermann, and A. Neumaier (eds.) Global Optimization and Constraint Satisfaction, vol. 2861 of Lecture Notes in Computer Science, pp. 123–141. Springer, Berlin/Heidelberg (2003)
37. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, London and New York (1969)
38. Qin, Z., Goldfarb, D., Ma, S.: An alternating direction method for total variation denoising, arXiv, preprint arXiv:1108.1587 (2011)

39. Toint, P.L.: Nonlinear stepsize control, trust regions and regularizations for unconstrained optimization. Optim. Methods Softw **28**, 82–95 (2013)
40. Wright, S.J.: Superlinear convergence of a stabilized SQP method to a degenerate solution. Comput. Optim. Appl. **11**, 253–275 (1998)
41. Yang, J., Zhang, Y., Yin, W.: A fast alternating direction method for tvl1-l2 signal reconstruction from partial fourier data. IEEE J. Sel. Top. Signal Proces. **4**, 288–297 (2010)